Eduardo
Bayro-Corrochano

Geometric Algebra Applications

Vol. II

Robot Modelling and Control

Springer

Geometric Algebra Applications Vol. II

Eduardo Bayro-Corrochano

Geometric Algebra Applications Vol. II

Robot Modelling and Control

Springer

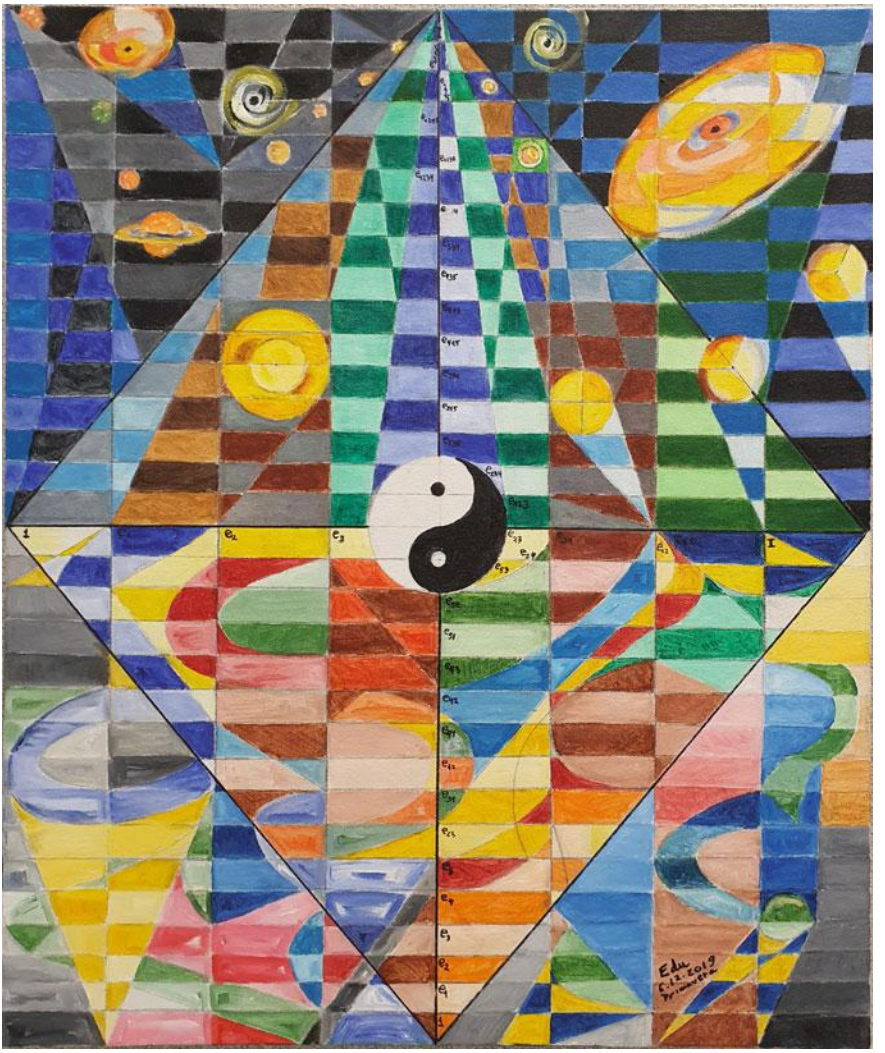
Eduardo Bayro-Corrochano
Electrical Engineering and Computer
Science Department
CINVESTAV
Guadalajara, Jalisco, Mexico

ISBN 978-3-030-34976-9 ISBN 978-3-030-34978-3 (eBook)
https://doi.org/10.1007/978-3-030-34978-3

© Springer Nature Switzerland AG 2020
This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.
The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.
The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, expressed or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

To my beloved wife Joanna Jablonska Bayro,
without her amorous and intelligent support
all these years rising our children
Nikolai, Claudio and Gladys,
this work would not have seen the light.



Dialectics of Space and Time. G_3–$G_{4,1}$ E. Bayro-Corrochano, January 2020

"The human relation with nature must be that of harmony and equilbrium. It needs to be dialetic, but without the destruction of nature as it would comprimise the proper existence of the human."
Dialectics of Nature, Friederich Engels, 1872–1882.

Preface

This book presents the theory and applications of an advanced mathematical language called *geometric algebra* that greatly helps to express the ideas and concepts and to develop algorithms in the broad domain of robot physics.

In the history of science, without essential mathematical concepts, theories would have not been developed at all. We can observe that in various periods of the history of mathematics and physics, certain stagnation occurred; from time to time, thanks to new mathematical developments, astonishing progress took place. In addition, we see that the knowledge became unavoidably fragmented as researchers attempted to combine different mathematical systems. Each mathematical system brings about some parts of geometry; however, together, these systems constitute a highly redundant system due to an unnecessary multiplicity of representations for geometric concepts. The author expects that due to his persistent efforts to bring to the community geometric algebra for applications as a metalanguage for geometric reasoning, in the near future tremendous progress in robotics should take place.

What is geometric algebra? Why is its application so promising? Why should researchers, practitioners, and students make the effort to understand geometric algebra and use it? We want to answer all these questions and convince the reader that becoming acquainted with geometric algebra for applications is a worthy undertaking.

The history of geometric algebra is unusual and quite surprising. In the 1870s, William Kingdon Clifford introduced his geometric algebra, building on the earlier works of Sir William Rowan Hamilton and Hermann Gunther Grassmann. In Clifford's work, we perceive that he intended to describe the geometric properties of vectors, planes, and higher dimensional objects. Most physicists encounter the algebra in the guise of Pauli and Dirac matrix algebras of quantum theory. Many roboticists or computer graphic engineers use quaternions for 3D rotation estimation and interpolation, as a pointwise approach is too difficult for them to formulate homogeneous transformations of high-order geometric entities. They resort often to tensor calculus for multivariable calculus. Since robotics and engineering make use of the developments of mathematical physics, many beliefs are automatically inherited; for instance, some physicists come away from a study of Dirac theory with the view that Clifford's algebra is inherently quantum mechanical. The goal of this

book is to eliminate these kinds of beliefs by giving a clear introduction of geometric algebra and showing this new and promising mathematical framework to multivectors and geometric multiplication in higher dimensions. In this new geometric language, most of the standard matter taught to roboticists and computer science engineers can be advantageously reformulated without redundancies and in a highly condensed fashion. The geometric algebra allows us to generalize and transfer concepts and techniques to a wide range of domains with little extra conceptual work. Leibniz dreamed of a geometric calculus system that deals directly with geometric objects rather than with sequences of numbers. It is clear that by increasing the dimension of the geometric space and the generalization of the transformation group, the invariance of the operations with respect to a reference frame will be more and more difficult. Leibniz's invariance dream is fulfilled for the n-D classical geometries using the coordinate-free framework of geometric algebra.

The aim of this book is precise and well planned. It is not merely an expose of mathematical theory; rather, the author introduces the theory and new techniques of geometric algebra, albeit by showing their applications in diverse domains ranging from neural computing and robotics to medical image processing.

Acknowledgements

Eduardo José Bayro-Corrochano would like to thank the Center for Research and Advanced Studies (CINVESTAV, Guadalajara, Mexico) and the Consejo Nacional de Ciencia y Tecnología (SEP-CONACYT, Mexico) for their support of this project. I am also very grateful to my former Ph.D. students Julio Zamora-Esquivel, Nancy Arana-Daniel, Carlos López-Franco, Jorge Rivera-Rovelo, Leo Reyes-Hendrick, Luis Eduardo Falcón, Carlos López-Franco, Rubén Machucho Cadena, Oscar Carbajal-Espinoza, Luis Enrique González-Jiménez, Guillermo Osuna-Gonzáles, and Jesús Medrano-Hermosillo for fruitful discussions and technical cooperation. Their creative suggestions, criticism, and patient research work were decisive for the completion of this book. In the geometric algebra community, first of all, I am indebted to David Hestenes for all his amazing work in developing the modern subject of geometric algebra and his constant encouragement to me for tackling problems in robot physics. Also, I am very thankful to Garret Sobczyk, John Selig, Eckard Hitzer, Dietmar Hildebrand, and Joan Lasenby for their support and constructive suggestions. Finally, I am very thankful to the Mexican people, who pay my salary, which made it possible for me to accomplish this contribution to scientific knowledge.

Guadalajara, Mexico Eduardo Bayro-Corrochano
January 2020

How to Use This Book

This section begins by briefly describing the organization and content of the chapters and their interdependency. Then, it explains how readers can use the book for self-study or for delivering a regular graduate course.

Chapter Organization

Chapter 1 presents the Geometric Algebra for Modeling in Robotic Physics. In this chapter, we explain the computational advantages of geometric algebra for modeling and solving problems in robotics, computer vision, artificial intelligence, neural computing, and medical robotics.

Part I Fundamentals of Geometric Algebra

Chapter 2 outlines the history of geometric algebra and gives an introduction to geometric algebra. After preliminary definitions, we explain multivector products and operations and we discuss how to handle linear algebra derivations and simplexes.

Chapter 3 examines Lie group theory, Lie algebra, and the algebra of incidence using the universal geometric algebra generated by reciprocal null cones. We start discussing the Lie groups and Lie algebras of rotors. We analyze the reciprocal null cones and study the General Linear Group as Spin Group. We study the $Pin^{sp}(3,3)$, the projective transformations using the null geometric algebras over $\mathbb{R}^{3,3}$ and the Lie groups in $\mathbb{R}^{3,3}$. We study the horosphere and n-dimensional affine space as well as the general linear Lie algebra $gl(\mathcal{N})$ of the general linear Lie group $GL(\mathcal{N})$. This chapter ends with the study of algebra of incidence and describes geometric constraints as flags.

Chapter 4 explains in detail the geometric algebras for 2D, 3D, and 4D spaces where the representation of geometric entities and their linear transformations using bivectors are explained.

Chapter 5 presents the kinematics of the 2D and 3D spaces. It describes the kinematics of points, lines, and planes using the Euclidean 3D geometric algebra and motor algebra.

Chapter 6 is devoted to the conformal geometric algebra. We explain the representation of geometric objects and versors. Different geometric configurations are studied using geometric objects as points, lines, planes circles, spheres, and simplexes. We briefly discuss the use of ruled surfaces.

Chapter 7 discusses advanced geometric algebras $G^{+}_{6,0,2}$, $G_{6,3}$ and $G_{9,3}$ and $G^{+}_{6,0,6}$.

Chapter 8 discusses the main issues of the implementation of computer programs for geometric algebra and how algorithms can be sped up using specialized hardware.

Part II Interpolation, Kinematics, and Dynamics

Chapter 9 describes the rigid motion interpolation of geometric entities using conformal geometric algebra.

Chapter 10 presents robot direct and inverse kinematics using lines and planes in motor algebra and using lines, planes, circles, and spheres in conformal geometric algebra.

Chapter 11 studies the robot dynamics using Euler–Lagrange in conformal geometric algebra and a recursive Newton–Euler method in terms of screw theory in the motor algebra framework. In this procedure, one can locally the Hamiltonians of the robot joints.

Part III Robot Control

Chapter 12 presents the modeling of the dynamics of robot manipulators using the Newton–Euler recursive algorithm, the computing of the local Hamiltonians, and the design of localized nonlinear controllers of robots using screw theory.

Chapter 13 applies the quaternion spike neural network for simultaneous system identification and control of robots.

Chapter 14 uses quadratic programming for the generation of references for the control and tracking of robots.

Part IV Applications I: Robot Vision, Quadrotor

Chapter 15 is devoted to line motion estimation using SVD and extended Kalman filter techniques.

Chapter 16 presents a tracker endoscope calibration and the calibration of sensors with respect to a robot frame. Here, we use purely a language of lines and motors.

Chapter 17 illustrates visual-guided grasping tasks using representations and geometric constraints developed using conformal geometric algebra.

Chapter 18 describes a 3D map reconstruction and relocalization using conformal geometric entities exploiting the Hough space.

Chapter 19 presents the dynamic modeling of a quadrotor using motor algebra and the application of nonlinear controllers for stability and tracking.

Part V Applications II: Medical Robotics

Chapter 20 studies the modeling and registration of medical data. It includes the application of marching spheres for 3D medical shape representation and registration.

Chapter 21 shows methods of geometric computing for minimal invasive surgery using the conformal geometric algebra.

Part VI Appendix

Chapter 22 presents the notation used in the book.

Chapter 23 includes useful formulas for geometric algebra used in the book and in the literature.

Interdependence of the Book Chapters

The interdependence of the book chapters is shown in Fig. 1. As introduction to the book, Chap. 1 starts with a broad explanation of the role of Geometric Algebra (GA) for modeling in robotic physics. Essentially, there are five groups of chapters.

- Part I Fundamentals of Geometric Algebra
  - Chapter 2: History and basic concepts of geometric algebra
  - Chapter 3: Lie algebras, Lie Groups, and algebra of incidence.
  - Chapters 4–6 useful GAs for 2D, 3D, and 4D and the conformal GA

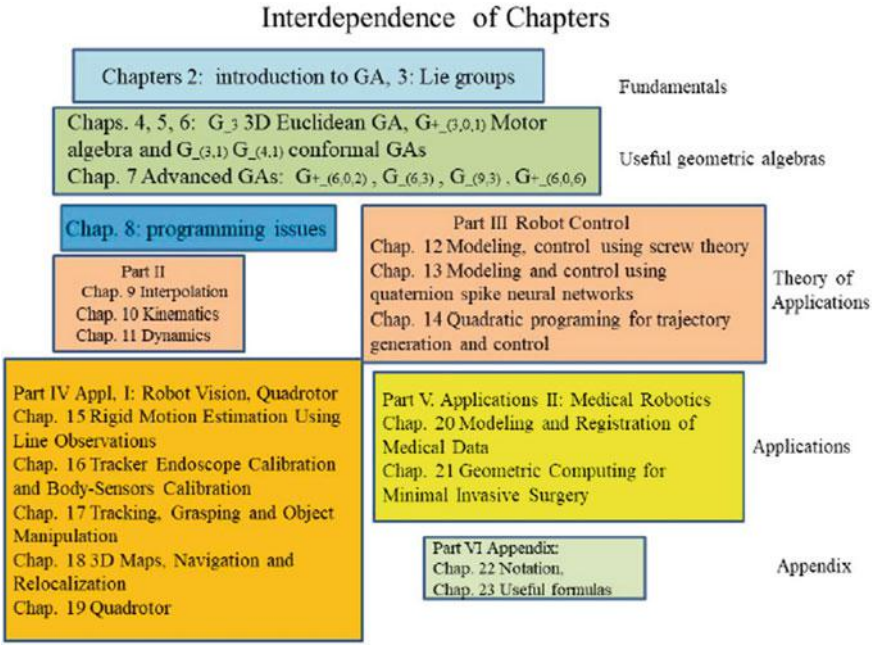

Interdependence of Chapters
Chapters 2: introduction to GA, 3: Lie groups
Fundamentals
Chaps. 4, 5, 6: G_3 3D Euclidean GA, G+_(3,0,1) Motor algebra and G_(3,1) G_(4,1) conformal GAs
Chap. 7 Advanced GAs: G+_(6,0,2) , G_(6,3) , G_(9,3) , G+_(6,0,6)
Useful geometric algebras
Chap. 8: programming issues
Part II
Chap. 9 Interpolation
Chap. 10 Kinematics
Chap. 11 Dynamics
Part III Robot Control
Chap. 12 Modeling, control using screw theory
Chap. 13 Modeling and control using quaternion spike neural networks
Chap. 14 Quadratic programing for trajectory generation and control
Theory of Applications
Part IV Appl. I: Robot Vision, Quadrotor
Chap. 15 Rigid Motion Estimation Using Line Observations
Chap. 16 Tracker Endoscope Calibration and Body-Sensors Calibration
Chap. 17 Tracking, Grasping and Object Manipulation
Chap. 18 3D Maps, Navigation and Relocalization
Chap. 19 Quadrotor
Part V. Applications II: Medical Robotics
Chap. 20 Modeling and Registration of Medical Data
Chap. 21 Geometric Computing for Minimal Invasive Surgery
Applications
Part VI Appendix:
Chap. 22 Notation.
Chap. 23 Useful formulas
Appendix

Fig. 1 Chapter Interdependence: Fundamentals → Theory of Applications → Applications → Appendix

  - Chapter 7 Advanced GAs $G^{+}_{6,0,2}, G^{+}_{6,3}, G^{+}_{9,3}, G^{+}_{6,0,6}$
  - Chapter 8 Programming issues

- Part II Interpolation, Kinematics, and Dynamics
  - Chapter 9 Interpolation of motion of geometric entities in conformal GA
  - Chapter 10 Direct and inverse kinematics using motor algebra and conformal GA
  - Chapter 11 Euler–Lagrange and recursive Newton–Euler using CGA y motor algebra

- Part III Robot Control
  - Chapter 12 Modeling and control of robot manipulators using screw theory
  - Chapter 13 Modeling and control using quaternion spike neural networks
  - Chapter 14 Quadratic programming for trajectory generation and control of robots

- Part IV Applications I: Robot Vision, Quadrotor
  - Chapter 15 Rigid Motion Estimation Using Line Observations
  - Chapter 16 Tracker Endoscope Calibration and Body-Sensor Calibration
  - Chapter 17 Tracking, Grasping, and Object Manipulation

  - Chapter 18 3D Maps, Navigation and Relocalization
  - Chapter 19 Quadrotor
- Part V Applications II: Medical Robotics
  - Chapter 20 Modeling and Registration of Medical Data
  - Chapter 21 Geometric Computing for Minimal Invasive Surgery
- Part VI Appendix
  - 22 Notation
  - 23 Useful formulas for Geometric Algebra

The reader should start with the Part I Fundamentals of Geometric Algebra, where Chaps. 2–7 provide very useful material to get an overall overview of the advantages of the geometric algebra framework and in Chap. 8 to know more about the key programming issues and consider our suggestions. For the more mathematically inclined readers, Chaps. 3 and 7 are recommended. Then select a domain of application of Parts I–V, read their theory, and get more insight by studying the related applications.

Audience

The book is aimed at a graduate level; thus, a basic knowledge of linear algebra and vector calculus is needed. A mind free of biases or prejudices caused by old-fashioned mathematics like matrix algebra and vector calculus is an even better prerequisite to reading this book.

Ideas and concepts in robot physics can be advantageously expressed using an appropriate language of mathematics. Of course, this language is not unique. There are many different algebraic systems, with their own advantages and disadvantages. Furthermore, treating a problem combining excessively different mathematical systems results in a fragmentation of the knowledge. In this book, we present geometric algebra, which we believe is the most powerful available mathematical system to date. It is a unifying language for treating robot physics; as a result, the knowledge is not fragmented and we can produce compact and less redundant mathematical expressions that are prone to be optimized for real-time applications.

One should simply start to formulate the problems using multivectors and versors and forget the formalism of matrices. However, bear in mind that when developing algorithms for particular problems, it is often a matter of techniques' integration. For example, one will model the problem with geometric algebra to find geometric constraints and optimize the resulting code for real-time execution; but it is also possible that one may resort for certain computations to other algorithms that were not necessarily developed using the geometric algebra framework, for

example, Cholesky decomposition or image preprocessing using a morphological filter.

No prior knowledge in computer vision, robotics, wavelets theory, or neural computing of graphics engineering is indispensable. If you are acquainted with topics of those fields, even better. The book is well suited for a taught course or for self-study at the postgraduate level. For the mathematically more inclined reader, Chaps. 3 and 7 are complementary. Readers wanting to know more about some definitions and concepts of the classical Clifford algebra should consult [11], Geometric Applications Vol. I, Chap. 20.

Exercises

The book offers 163 exercises, so that the reader can gradually learn to formulate equations and compute by hand or using a software package. The exercises were formulated to teach the fundamentals of geometric algebra and also to stimulate the development of the readers' skills in creative and efficient geometric computing.

Chapter 2 includes 35 exercises for basic geometric algebra computations to learn to get along with the generalized inner product, wedge of multivectors, and some exercises in differential multivector calculus.

Chapter 3 presents 32 exercises related to Lie group theory, Lie algebra, and the algebra of incidence using the reciprocal null cones of the universal geometric algebra, directed distance, and the application of flags in the affine plane. Includes the computation of versors of the 3D projective group

Chapter 4 offers 19 exercises for 2D, 3D, and 4D geometric algebras, particularly to get acquainted with Lie and bivector algebras, power series involving the exterior exponent, and rotor and motor operations for practical use.

Chapter 5 has 20 exercises related to the 2D and 3D kinematics of points, lines, and planes using 3D and 4D geometric algebras. It demands exercises about bivector relations, motion of geometric entities, velocity using motors, and flags for describing configurations of geometric entities attached to an object frame.

Chapter 6 provides 35 exercises, consisting of basics computations with versor techniques and geometric entities as the point, line, plane, circle, planes, spheres, flats, plunges, and carriers, and computations in the dual space. There are also exercises to prove classical theorems in conformal geometric algebra.

Chapter 7 presents the geometric algebras $G^{+}_{6,0,2}$, $G_{6,3}$, $G^{+}_{9,3}$, $G^{+}_{6,0,6}$ an it provides 22 exercises to be solved in these mathematical frameworks like modeling sphere, ellipsoids, conoids, cylinders, hyperboloids, and their kinematics; topics in geometry like line geometry, screw and co-screws, twists and wrenches, kinematics and dynamics.

Use of Computer Geometric Algebra Programs

In recent years, we have been using different software packages to check equations, prove theorems, and develop algorithms for applications.

We recommend using CLICAL for checking equations and proving theorems. CLICAL was developed by Pertti Lounesto's group and is available at: http://users.tkk.fi/ppuska/mirror/Lounesto/CLICAL.htm.

For proving equations using symbolic programming, we recommend using the Maple-based package CLIFFORD of Rafal Ablamowicz, which supports $N \leq 9$ and is available at: http://math.tntech.edu/rafal.

For practicing and learning geometric algebra computing, and to try a variety of problems in computer science and graphics, we suggest using the *C++*-based CLUCal of Christian Perwass, available at: http://www.perwass.de/cbup/clu.html and GAIGEN2 of Leo Dorst's group, which generates fast *C++* or JAVA sources for low-dimensional geometric algebra, available at: http://www.science.uva.nl/ga/gaigen/.

A very powerful multivector software for applications in computer science and physics is the *C++* MV 1.3.0 to 1.6 sources supporting $N \leq 63$ of Ian Bell which is available at: http://www.iancgbell.clara.net/maths/index.htm.

The reader can also download our *C++* and MATLAB programs, which are being routinely used for applications in robotics, image processing, wavelets transforms, computer vision, neural computing, and medical robotics. These are available at: http://www.gdl.cinvestav.mx/edb/GAprogramming.

Use as a Textbook

In the last decade, the author has taught in various universities a postgraduate course on the topic of geometric algebra and its applications: at the Institut für Informatik und Angewandte Mathematik der Christian Albrechts Universität (WS 1998), Kiel Germany, CINVESTAV Department of Electrical Engineering and Computer Science (annually since 2001) in Guadalajara, Mexico, and at the Informatyk Institut der Karlsruhe Universität (SS 2008, 2 SWS, LV.- No. 24631), in Karlsruhe, Germany. The author has been involved with his collaborators in developing concepts and computer algorithms for a variety of application areas. At the same time, he has established a course on geometric algebra and its applications that has proven to be accessible and very useful for beginners. In general, the author follows a sequence of topics based on the material in this book. From the very beginning, students use software packages to try what is taught in the classroom.

- Chapter 2 gives an introduction to geometric algebra.
- Chapter 3 presents Lie algebras, Lie groups of rotors, geometric algebra of the reciprocal null cones, the general linear group as spin group, $Pin^{sp}(3,3)$, the horosphere and the affine plane, the general linear Lie algebra $gl(N)$, and the general linear Lie group and the algebra of Incidence.

- Chapter 4 begins explaining the geometric algebra models for 2D, 3D, and 4D. This allows the student to learn to represent linear transformations as rotors, represent points, lines, and planes, and apply rotors and translators using these geometric entities. The duality principle is explained.
- Chapter 5 presents the kinematics of points, lines, and planes using 3D geometric algebra and motor algebra. The big step here is the use of homogeneous transformations to linearize the rigid motion transformation. Once the student has understood motor algebra, he or she has become proficient in modeling and applying a language of points, lines, and planes to tackle various problems in robotics, computer vision, and mechanical engineering.
- After these experiences, the student should move to treating projective geometry in the 4D geometric algebra $G_{3,1}$ (Minkowski metric). The role of the projective split will be explained. If the student has attended a lecture on computer vision where n-view geometry is taught [67], it is a big advantage to better understand the use of geometric algebra for projective geometry, see [11], Geometric Applications Vol. I, Chap. 12.
- Chapter 6 is devoted to conformal geometric algebra, where the experiences gained with motor algebra and $G_{3,1}$ for projective geometry are key to understanding homogeneous representations, null vectors, and the use of the null cone in the conformal geometric algebra framework. The role of the conformal additive and multiplicative splits is explained. The representation of geometric objects and versors is also shown. Different geometric configurations are studied, including simplexes, flats, plunges, and carriers. We briefly discuss the promising use of ruled surfaces and the role of conformal mapping in computer vision.
- Chapter 7 offers the advanced geometric algebras $G^{+}_{6,0,2}$, $G_{6,3}$, $G^{+}_{9,3}$, $G^{+}_{6,0,6}$. Since in these geometric algebras you have more degrees of freedom, namely, bigger multivector basis, so you can in these mathematical frameworks model spheres, ellipsoids, conoids, cylinders, hyperboloids, and their kinematics; topics in geometry like line geometry, screw and co-screws, twists and wrenches, kinematics and dynamics, and surprisingly also model as spinors the tensors of affine and projective transformations like inertia tensors or projective transformations which in lower dimensional GAs are not possible.

In each theme, the lecturer illustrates the theory with real applications in the domains of computer vision, robotics, control engineering, robot vision, neural computing, medical robotics, and graphic engineering. For that, the book presents illustrations of real applications: Part III Robot Control, Part IV Applications I: Robot Vision and Quadrotor, and Part V Applications II: Medical Robotics.

The analysis of the application examples is a good practice for the student, as they will go through well-designed training illustrations. In this way, they will gain more insight into the potential of geometric algebra, learn how it has to be applied, and finally get algorithms that are directly applicable in their research and engineering projects.

If lecturers around the world want to hold a course on geometric algebra and its applications, they can use this book and a PowerPoint presentation of our lecturers, which are available for free use at: http://www.gdl.cinvestav.mx/edb/GAlecture.

Contents

1 Geometric Algebra for Modeling in Robotic Physics 1
1.1 The Roots of Geometry and Algebra 1
1.2 Geometric Algebra a Unified Mathematical Language 3
1.3 What Does Geometric Algebra Offer for Geometric Computing? 4
1.3.1 Coordinate-Free Mathematical System 4
1.3.2 Models for Euclidean and Pseudo-Euclidean Geometry 5
1.3.3 Subspaces as Computing Elements 7
1.3.4 Representation of Orthogonal Transformations 7
1.3.5 Objects and Operators 7
1.3.6 Extension of Linear Transformations 9
1.3.7 Signals and Wavelets in the Geometric Algebra Framework 9
1.3.8 Kinematics and Dynamics 10
1.4 Solving Problems in Perception and Action Systems 10
References 16

Part I Fundamentals of Geometric Algebra

2 Introduction to Geometric Algebra 21
2.1 History of Geometric Algebra 21
2.2 What Is Geometric Algebra? 23
2.2.1 Basic Definitions 23
2.2.2 Nonorthonormal Frames and Reciprocal Frames 26
2.2.3 Reciprocal Frames with Curvilinear Coordinates 27
2.2.4 Some Useful Formulas 28
2.3 Multivector Products 29
2.3.1 Further Properties of the Geometric Product 31
2.3.2 Projections and Rejections 32

2.3.3 Projective Split . . . 33
2.3.4 Generalized Inner Product . . . 34
2.3.5 Geometric Product of Multivectors . . . 36
2.3.6 Contractions and the Derivation . . . 37
2.3.7 Hodge Dual . . . 39
2.3.8 Dual Blades and Duality in the Geometric Product . . . 40
2.4 Multivector Operations . . . 41
2.4.1 Involution, Reversion, and Conjugation Operations . . . 41
2.4.2 Join and Meet Operations . . . 43
2.4.3 Multivector-Valued Functions and the Inner Product . . . 44
2.4.4 The Multivector Integral . . . 45
2.4.5 Convolution and Correlation of Scalar Fields . . . 45
2.4.6 Clifford Convolution and Correlation . . . 46
2.5 Linear Algebra . . . 47
2.5.1 Linear Algebra Derivations . . . 48
2.6 Simplexes . . . 48
2.7 Exercises . . . 50
References . . . 54

3 Lie Algebras, Lie Groups, and Algebra of Incidence . . . 55
3.1 Introduction . . . 55
3.2 Lie Algebras and Bivector Algebras . . . 55
3.3 Lie Groups and Lie Algebras of Rotors . . . 56
3.3.1 Lie Group of Rotors . . . 56
3.3.2 Bivector Algebra . . . 58
3.4 Complex Structures and Unitary Groups . . . 59
3.4.1 The Doubling Bivector . . . 60
3.4.2 Unitary Groups $U(n)$ and Unitary Lie Algebras $u(n)$ . . . 61
3.5 Geometric Algebra of Reciprocal Null Cones . . . 63
3.5.1 Reciprocal Null Cones . . . 63
3.5.2 The Universal Geometric Algebra $G_{n,n}$ . . . 65
3.5.3 Representations and Operations Using Bivector Matrices . . . 65
3.5.4 Bivector Representation of Linear Operators . . . 67
3.5.5 The Standard Bases of $G_{n,n}$ . . . 68
3.5.6 Endomorphisms of $G_{n,n}$ . . . 70
3.5.7 The Bivectors of $G_{n,n}$ . . . 72
3.5.8 Mappings Acting on the Null Space . . . 74
3.5.9 The Bivector or Lie Algebra of $G_{n,n}$ . . . 75
3.6 The General Linear Group as Spin Group . . . 75

3.6.1 The Special Linear Group SL($n, \mathbb{R}$) . . . 78
3.6.2 The Pin(n) and Spin(n) Groups . . . 79
3.7 The $Pin^{sp}(3,3)$ . . . 81
3.7.1 Line Geometry Using Null Geometric Algebra Over $\mathbb{R}^{3,3}$ . . . 82
3.7.2 Projective Transformations Using Null Geometric Algebra Over $\mathbb{R}^{3,3}$ . . . 83
3.7.3 Lie Groups in $\mathbb{R}^{3,3}$ . . . 85
3.8 2D Projective Geometry in $\mathbb{R}^{3,3}$ . . . 93
3.9 Horosphere and n-Dimensional Affine Plane . . . 94
3.9.1 The Horosphere . . . 94
3.9.2 The n-Dimensional Affine Plane . . . 95
3.10 The General Linear Algebra $gl(\mathcal{N})$ of the General Linear Lie Group $GL(\mathcal{N})$ . . . 96
3.10.1 The General Linear Lie Group $GL(\mathcal{N})$ . . . 96
3.10.2 The General Linear Algebra $gl(\mathcal{N})$ . . . 99
3.10.3 The Orthogonal Groups . . . 100
3.10.4 Computing Rigid Motion in the Affine Plane . . . 102
3.10.5 The Lie Algebra of the Affine Plane . . . 103
3.11 The Algebra of Incidence . . . 108
3.11.1 Incidence Relations in the Affine n-Plane . . . 110
3.11.2 Directed Distances . . . 111
3.11.3 Incidence Relations in the Affine 3-Plane . . . 112
3.11.4 Geometric Constraints as Flags . . . 114
3.12 Conclusion . . . 114
3.13 Exercises . . . 115
References . . . 122

4 2D, 3D, and 4D Geometric Algebras . . . 125
4.1 Complex, Double, and Dual Numbers . . . 125
4.2 2D Geometric Algebras of the Plane . . . 126
4.3 3D Geometric Algebra for the Euclidean 3D Space . . . 128
4.3.1 The Algebra of Rotors . . . 129
4.3.2 Orthogonal Rotors . . . 133
4.3.3 Recovering a Rotor . . . 134
4.4 Quaternion Algebra . . . 135
4.5 4D Geometric Algebra for 3D Kinematics . . . 137
4.5.1 Motor Algebra . . . 138
4.5.2 Motors, Rotors, and Translators in $G^{+}_{3,0,1}$ . . . 139
4.5.3 Properties of Motors . . . 143
4.5.4 The Klein Manifold . . . 144
4.5.5 Reciprocal Screws . . . 145
4.5.6 The Study Manifold . . . 146

4.6 4D Geometric Algebra for Projective 3D Space 147
4.7 Conclusion 148
4.8 Exercises 148
References 151

5 Kinematics of the 2D and 3D Spaces 153
5.1 Introduction 153
5.2 Representation of Points, Lines, and Planes Using 3D Geometric Algebra 153
5.3 Representation of Points, Lines, and Planes Using Motor Algebra 155
5.4 Representation of Points, Lines, and Planes Using 4D Geometric Algebra 156
5.5 Motion of Points, Lines, and Planes in 3D Geometric Algebra 157
5.6 Motion of Points, Lines, and Planes Using Motor Algebra . . . 158
5.7 Motion of Points, Lines, and Planes Using 4D Geometric Algebra 160
5.8 Spatial Velocity of Points, Lines, and Planes 161
5.8.1 Rigid Body Spatial Velocity Using Matrices 161
5.8.2 Angular Velocity Using Rotors 166
5.8.3 Rigid Body Spatial Velocity Using Motor Algebra . . . 168
5.8.4 Point, Line, and Plane Spatial Velocities Using Motor Algebra 170
5.9 Differential Kinematics 171
5.10 Incidence Relations Between Points, Lines, and Planes 174
5.10.1 Flags of Points, Lines, and Planes 174
5.11 Conclusion 175
5.12 Exercises 175
References 179

6 Conformal Geometric Algebra 181
6.1 Introduction 181
6.1.1 Conformal Split 182
6.1.2 Conformal Splits for Points and Simplexes 183
6.1.3 Euclidean and Conformal Spaces 185
6.1.4 Stereographic Projection 187
6.1.5 Inner and Outer Product Null Spaces 189
6.1.6 Spheres and Planes 190
6.1.7 Geometric Identities, Dual, Meet, and Join Operations 192
6.1.8 Simplexes and Spheres 196
6.2 The 3D Affine Plane 197

6.2.1 Lines and Planes . . . 198
6.2.2 Directed Distance . . . 199
6.3 The Lie Algebra . . . 199
6.4 Conformal Transformations . . . 201
6.4.1 Inversion . . . 202
6.4.2 Reflection . . . 204
6.4.3 Translation . . . 204
6.4.4 Transversion . . . 205
6.4.5 Rotation . . . 205
6.4.6 Rigid Motion Using Flags . . . 206
6.4.7 Dilation . . . 208
6.4.8 Involution . . . 208
6.4.9 Conformal Transformation . . . 209
6.5 Ruled Surfaces . . . 209
6.5.1 Cone and Conics . . . 209
6.5.2 Cycloidal Curves . . . 210
6.5.3 Helicoid . . . 211
6.5.4 Sphere and Cone . . . 212
6.5.5 Hyperboloid, Ellipsoids, and Conoid . . . 212
6.6 Exercises . . . 212
References . . . 216

7 The Geometric Algebras $\mathbf{G}^{+}_{6,0,2}$, $\mathbf{G}_{6,3}$, $\mathbf{G}^{+}_{9,3}$, $\mathbf{G}^{+}_{6,0,6}$ . . . 219
7.1 Introduction . . . 219
7.2 The Double Motor Algebra $G^{+}_{6,0,2}$ . . . 220
7.2.1 The Shuffle Product . . . 223
7.2.2 Equations of Motion . . . 224
7.3 The Geometric Algebra $G_{6,3}$ . . . 228
7.3.1 Additive Split of $G_{6,3}$ . . . 229
7.3.2 Geometric Entities of $G_{6,3}$ . . . 231
7.3.3 Intersection of Surfaces . . . 233
7.3.4 Transformations of $G_{6,3}$ . . . 235
7.4 The Geometric Subalgebras $G^{+}_{9,3}$ and $G^{+}_{6,0,6}$ . . . 236
7.5 Exercises . . . 240
References . . . 242

8 Programming Issues . . . 243
8.1 Main Issues for an Efficient Implementation . . . 243
8.1.1 Specific Aspects for the Implementation . . . 244
8.2 Implementation Practicalities . . . 245

8.2.1 Specification of the Geometric Algebra $G_{p,q}$ 245
8.2.2 The General Multivector Class 245
8.2.3 Optimization of Multivector Functions 246
8.2.4 Factorization 247
8.2.5 Speeding up Geometric Algebra Expressions 248
8.2.6 Multivector Software Packets 249
8.2.7 Specialized Hardware to Speed up Geometric Algebra Algorithms 251
References 251

Part II Interpolation, Kinematics, and Dynamics

9 Rigid Motion Interpolation 255
9.1 The Motor Spherical Linear Interpolation Function 255
9.2 Study Quadric Interpolation Algorithm 257
9.2.1 Interpolation Algorithm 257
9.2.2 Motor Interpolation 258
9.3 GPU Acceleration 260
9.3.1 Methodology 261
9.3.2 Standard GPU Implementation 262
9.3.3 Multi-streaming Implementation 262
9.4 Experimental Results 263
9.4.1 Serial and Parallel Speed-Up Comparison 265
9.4.2 Multi-streaming Implementation 265
9.5 Conclusions 266
References 266

10 Robot Kinematics 269
10.1 Introduction 269
10.2 Screw Theory 270
10.2.1 Reciprocal Screws 270
10.3 Elementary Transformations of Robot Manipulators 270
10.3.1 The Denavit–Hartenberg Parameterization 271
10.3.2 Representations of Prismatic and Revolute Transformations 273
10.3.3 Grasping by Using Constraint Equations 275
10.4 Direct Kinematics of Robot Manipulators 277
10.4.1 Maple Program for Motor Algebra Computations 278
10.5 Inverse Kinematics of Robot Manipulators Using Motor Algebra 279
10.5.1 The Rendezvous Method 280
10.5.2 Computing θ_1, θ_2, and d_3 Using a Point 280
10.5.3 Computing θ_4 and θ_5 Using a Line 284
10.5.4 Computing θ_6 Using a Plane Representation 286

10.6 Inverse Kinematic Using the 3D Affine Plane . . . 287
10.7 Inverse Kinematic Using Conformal Geometric Algebra . . . 290
10.8 Conclusion . . . 294
References . . . 295

11 Robot Dynamics . . . 297
11.1 Introduction . . . 297
11.2 D'Alembert's Principle . . . 297
11.3 Newtonian Mechanics . . . 298
11.3.1 Screw and Twist . . . 301
11.3.2 Momentum and Wrench . . . 303
11.3.3 Compliance . . . 305
11.3.4 Moments of Inertia . . . 306
11.3.5 Rigid Transformation of the Inertia . . . 307
11.3.6 Time Derivatives of the Inertia . . . 309
11.4 Newton–Euler Recursive Algorithm for Serial Robot Manipulators . . . 312
11.4.1 Single Robot Link . . . 312
11.4.2 Two Robot Links . . . 314
11.4.3 N Robot Links . . . 316
11.4.4 Recursive Algorithm . . . 318
11.5 Euler–Lagrange Dynamics . . . 321
11.5.1 Kinetic Energy . . . 321
11.5.2 Potential Energy . . . 328
11.5.3 Lagrange's Equations . . . 329
11.5.4 Computing $\mathcal{M}$ . . . 337
11.5.5 Computing G . . . 337
11.6 Constrained Variational Problems . . . 338
11.6.1 Optimization . . . 339
11.7 Euler–Lagrange Multivector Equations . . . 342
11.8 Hamiltonian Mechanics on Phase Space . . . 343
11.8.1 Symplectic Structure on Vector Manifolds . . . 343
11.8.2 The Hamiltonian and the Poisson Bracket . . . 344
11.9 Integral Invariants . . . 350
11.9.1 Lagrangian and Hamiltonian for Electromechanical or Robot Systems . . . 352
11.10 Iterative Computing of the Local Hamiltonian and the Derivative of the Momenta . . . 354
11.11 Conclusion . . . 356
References . . . 357

Part III Robot Control

12 Control of Robot Manipulators 361
12.1 Robot Controllers Using Screw Theory 361
12.2 Experimental Analysis of Controllers Using Screw Theory . . . 364
12.3 Robot Controllers Using Hamiltonians 367
12.3.1 Localized Hamiltonian Bang–Bang Control 368
12.4 PD and Sliding Mode Controllers for Localized Hamiltonian Control 369
12.5 Experimental Analysis of Controllers Using the Robot Hamiltonians 372
12.5.1 Bang–Bang Control 372
12.5.2 PD and Sliding Mode Control 373
12.6 Conclusion 376
References 377

13 Robot Neurocontrol 379
13.1 Introduction 379
13.2 Quaternion Spiking Neural Networks 379
13.2.1 Neuronal Model 380
13.2.2 Neuronal Architecture 382
13.3 Training Algorithm 383
13.3.1 Rotation with Quaternions 384
13.3.2 Quaternion Spike Neural Network 386
13.4 Control of a Simulated Nonlinear System 386
13.5 Control of a Real Nonlinear System 391
13.5.1 Forward Kinematics 391
13.5.2 The Jacobian 393
13.5.3 Inverse Kinematics 396
13.5.4 Control 397
13.6 Neural Network Signal Processing 400
13.6.1 Myo Bracelet 402
13.6.2 Physiology of Robotic Prosthesis 402
13.6.3 Preprocessing and Training 406
13.6.4 Evaluation and Control 407
13.7 Control of a Hand Prosthesis 408
13.8 Conclusions 411
References 412

14 Robot Control and Tracking 413
14.1 Introduction 413
14.2 Optimization Using Geometric Constraints 413
14.2.1 Desired Motion 413
14.2.2 Collision Avoidance 415

14.2.3 Quadratic Programming . . . 417
14.2.4 Control . . . 418
14.3 Integral Sliding Mode Control . . . 420
14.4 Experimental Analysis . . . 422
14.4.1 Parameters of the Dynamic Model . . . 422
14.4.2 Integral Sliding Mode Controller . . . 423
14.5 Conclusion . . . 426
References . . . 427

Part IV Applications I: Robot Vision, Quadrotor

15 Rigid Motion Estimation Using Line Observations . . . 431
15.1 Introduction . . . 431
15.2 Batch Estimation Using SVD Techniques . . . 431
15.2.1 Solving $AX = XB$ Using Motor Algebra . . . 433
15.2.2 Estimation of the Hand–Eye Motor Using SVD . . . 436
15.3 Experimental Results . . . 438
15.4 Discussion . . . 439
15.5 Recursive Estimation Using Kalman Filter Techniques . . . 441
15.5.1 The Kalman Filter . . . 441
15.5.2 The Extended Kalman Filter . . . 443
15.5.3 The Rotor Extended Kalman Filter . . . 445
15.6 The Motor Extended Kalman Filter . . . 448
15.6.1 Representation of the Line Motion Model in Linear Algebra . . . 449
15.6.2 Linearization of the Measurement Model . . . 451
15.6.3 Enforcing a Geometric Constraint . . . 452
15.6.4 Operation of the MEKF Algorithm . . . 454
15.6.5 Estimation of the Relative Positioning of a Robot End-Effector . . . 457
15.7 Conclusion . . . 460
References . . . 462

16 Tracker Endoscope Calibration and Body-Sensor Calibration . . . 463
16.1 Device-Camera Calibration . . . 463
16.1.1 Rigid Body Motion in CGA . . . 463
16.1.2 Hand–Eye Calibration in CGA . . . 465
16.1.3 Tracker Endoscope Calibration . . . 467
16.2 Body-Sensor Calibration . . . 469
16.2.1 Body–Eye Calibration . . . 470
16.2.2 Algorithm Simplification . . . 473
16.3 Conclusions . . . 475
References . . . 476

17 Tracking, Grasping, and Object Manipulation 477
17.1 Tracking 477
17.1.1 Exact Linearization via Feedback 478
17.1.2 Visual Jacobian 480
17.1.3 Exact Linearization via Feedback 482
17.1.4 Experimental Results 482
17.2 Barrett Hand Direct Kinematics 483
17.3 Pose Estimation 486
17.3.1 Segmentation 486
17.3.2 Object Projection 488
17.4 Grasping Objects 491
17.4.1 First Style of Grasping 491
17.4.2 Second Style of Grasping 493
17.4.3 Third Style of Grasping 494
17.5 Target Pose 495
17.5.1 Object Pose 496
17.6 Visually Guided Grasping 497
17.6.1 Results 498
17.7 Fuzzy Logic and Conformal Geometric Algebra for Grasping 498
17.7.1 Mandami Fuzzy System 500
17.7.2 Direct Kinematics of the Barrett Hand 500
17.7.3 Fussy Grasping of Objects 502
17.8 Conclusion 503
References 504

18 3D Maps, Navigation, and Relocalization 507
18.1 Map Building 507
18.1.1 Matching Laser Readings 507
18.1.2 Map Building 510
18.1.3 Line Map 510
18.1.4 3D Map Building 512
18.2 Navigation 513
18.2.1 Localization 514
18.2.2 Adding Objects to the 3D Map 514
18.2.3 Path Following 514
18.3 3D Map Building Using Laser and Stereo Vision 519
18.3.1 Laser Rangefinder 522
18.3.2 Stereo Camera System with Pan–Tilt Unit 525
18.4 Relocation Using Lines and the Hough Transform 526
18.5 Experiments 529
18.6 Conclusions 529
References 530

19 Quadrotor 531
19.1 Introduction 531
19.2 Rigid Body Spatial Velocity Using Motor Algebra 532
19.3 Mathematical Model Based on Motor Algebra to Multi-copter 533
19.4 Control Design 535
19.5 Simulation Results 539
19.6 Conclusion 542
References 542

Part V Applications II: Medical Robotics

20 Modeling and Registration of Medical Data 547
20.1 Background 547
20.1.1 Union of Spheres 547
20.1.2 The Marching Cubes Algorithm 548
20.2 Segmentation 550
20.3 Marching Spheres 553
20.3.1 Experimental Results for Modeling 555
20.4 Registration of Two Models 557
20.4.1 Sphere Matching 558
20.4.2 Experimental Results for Registration 560
20.5 Conclusions 562
References 563

21 Geometric Computing for Minimal Invasive Surgery 565
21.1 Tracker Endoscope Calibration 567
21.2 Planning Surgical Paths in Virtual World 568
21.3 Execution of Surgical Maneuvers 569
21.4 Registration of the Organ 571
21.5 Representing the Elasticity on the Organ Surface 572
21.6 Procedure for Surgery Cuts 573
21.6.1 Handling the US Probe 576
21.6.2 Suture 579
21.7 Conclusions 582
References 583

Appendix A: Notation 585

Appendix B: Useful Formulas for Geometric Algebra 587

Index 593

Chapter 1
Geometric Algebra for Modeling in Robotic Physics

> Scientists explore the physical world for reproducible patterns, which they represent by models and organize into theories according to laws.
> David Orlin Hestenes (1933–∞)

In this chapter, we will discuss the advantages for geometric computing that geometric algebra offers for solving problems and developing algorithms in the fields of artificial intelligence, robotics, and intelligent machines acting within the perception and action cycle. We begin with a short tour of the history of mathematics to find the roots of the fundamental concepts of geometry and algebra.

1.1 The Roots of Geometry and Algebra

The lengthy and intricate road along the history of mathematics shows that the evolution through time of the two domains *algebra* from the Arabic "alg-jbar" and geometry from the ancient Greek $\gamma\varepsilon\omega\mu\varepsilon\tau\rho\acute{\iota}\alpha$ (geo = earth, $metria$ = measure) started to intermingle early, depending upon certain trends imposed by the different groups and schools of mathematical thought. It was only at the end of the nineteenth century that they become a sort of a clear, integrated mathematical system.

Broadly speaking, on the one hand, algebra is a branch of mathematics concerning the study of structure, relation, and quantity. In addition, to work in algebra, just directly with numbers, algebra covers the work involving symbols, variables, and set elements. Addition and multiplication are considered general operations, which, in a more general view, lead to mathematical structures such as groups, rings, and fields.

On the other hand, geometry is concerned with essential questions of size, shape, and relative position of figures and with properties of space. Geometry is one of the oldest sciences initially devoted to practical knowledge concerned with lengths, areas, and volumes. In the third century, Euclid put geometry in an axiomatic form, and Euclidean geometry was born. During the first half of the seventeenth century, René Descartes introduced coordinates, and the concurrent development of algebra

© Springer Nature Switzerland AG 2020

E. Bayro-Corrochano, *Geometric Algebra Applications Vol. II*,
https://doi.org/10.1007/978-3-030-34978-3_1

evolved into a new stage of geometry, because figures such as plane curves could now be represented analytically with functions and equations. In the 1660s, G. Leibniz and Newton, both inventors of infinitesimal calculus, pursued a geometric calculus for dealing with geometric objects rather than with sequences of numbers. The analysis of the intrinsic structure of geometric objects with the works of Euler and Gauss further enriched the topic of geometry and led to the creation of topology and differential geometry. Since the nineteenth-century discovery of non-Euclidean geometry, the traditional concept of space has undergone a profound and radical transformation. Contemporary geometry postulates the concept of manifolds, spaces that are greatly more abstract than the classical Euclidean space and that approximately look alike at small scales. These spaces endowed an additional structure: a differentiable structure that allows one to do calculus, that is, a Riemannian metric allows us to measure angles and distances, symplectic manifolds serve as the phase spaces in the Hamiltonian formalism of classical mechanics, and the 4D Lorentzian manifolds serve as a space–time model in general relativity.

Algebraic geometry is considered a branch of the mathematics that combines techniques of abstract algebra with the language and problems of geometry. It plays a central role in modern mathematics and has multiple connections with a variety of fields: complex analysis, topology, and number theory. Algebraic geometry is fundamentally concerned with the study of algebraic varieties that are geometric manifestations of solutions of systems of polynomial equations. One looks for the Gröbner basis of an ideal in a polynomial ring over a field. The most studied classes of algebraic varieties are the *plane*, *algebraic curves* such as lines, parabolas, lemniscates, and Cassini ovals. Most of the developments of algebraic geometry in the twentieth century were within an abstract algebraic framework studying the intrinsic properties of algebraic properties independent of a particular way of embedding the variety in a setting of a coordinated space as in topology and complex geometry. A key distinction between projective geometry and algebraic geometry is that the former is more concerned with the more geometric notion of the point, whereas the latter puts major emphasis on the more analytical concepts of a regular function and a regular map and extensively draws on sheaf theory.

In the field of mathematical physics, geometric algebra is a multilinear algebra described more technically as a Clifford algebra that includes the geometric product. As a result, the theory, axioms, and properties can be built up in a more intuitive and geometric way. Geometric algebra is a coordinate-free approach to geometry based on the algebras of Grassmann [1] and Clifford [2]. Since the 1960s, Hestenes [3] has contributed to developing geometric algebra as a unifying language for mathematics and physics [4, 5]. Hestenes also presented a study of projective geometry using Clifford algebra [6] and recently the essential concepts of conformal geometric algebra [7]. David Hestenes summarized and tightened the role of algebra and geometry in a profound comment emphasizing the role of the capacities of language and spatial perception of the human mind, which, in fact, is the goal of this section. We reproduce it to finalize the section as a prelude to the next section to motivate and justify why geometric algebra can be of great use to build the intelligent machine of which Turing dreamed.

> In his famous survey of mathematical ideas, F. Klein championed the fusing of arithmetic with geometry as a major unifying principle of mathematics. Klein's seminal analysis of the structure and history of mathematics brings to light two major processes by which mathematics grows and becomes organized. They may be aptly referred to as the algebraic and geometric. The one emphasizes algebraic structure, while the other emphasizes geometric interpretation. Klein's analysis shows one process alternatively dominating the other in the historical development of mathematics. But there is no necessary reason that the two processes should operate in mutual exclusion. Indeed, each process is undoubtedly grounded in one of two great capacities of the human mind: the capacity for language and the capacity for spatial perception. From the psychological point of view, then, the fusion of algebra with geometry is so fundamental that one could say, Geometry without algebra is dumb! Algebra without geometry is blind!
>
> —D. Hestenes, 1984

1.2 Geometric Algebra a Unified Mathematical Language

First of all, let us analyze the problems of the community when they use classical mathematical systems to tackle problems in physics or robotics. In this regard, let us resort to an enlightening paper of Hestenes [8] where the author discusses the main issues for modeling the physical reality. The invention of analytical geometry and calculus was essential for Newton to create classical mechanics. On the other hand, the invention of tensor analysis was essential for Einstein to create the theory of relativity. The point here is that without essential mathematical concepts, both theories would not have been developed at all. We can observe in some periods of the history of mathematics certain stagnation and from time to time, thanks to new mathematical developments, astonishing progress in diverse fields. Furthermore, we can notice that as researchers attempt to combine different mathematical systems, unavoidably this attitude in research leads to a fragmentation of knowledge. Each mathematical system brings about some parts of geometry; however, together they constitute a highly redundant system, that is, an unnecessary multiplicity of representations for geometric concepts, see Fig. 1.2. This approach in mathematical physics and in robotics has the following pressing defects:

- Restricted access: The ideas, concepts, methods, and results are unfortunately disseminated across the mathematical systems. Being proficient only in a few of the systems, one does not have access to knowledge formulated in terms of the other mathematical systems.
- Redundancy: Less efficiency due to the repetitive representation of information in different mathematical systems.
- Deficient integration/articulation: Incoherences, incongruences, lack of generalizations, and ineffective integration and articulation of the different mathematical systems.
- Hidden common knowledge: Intrinsic properties and relations represented in different symbolic systems are very difficult to handle.

– Low information density: The information of the problem in question is reduced due to distribution over different symbolic systems.

According to Hestenes [8], the development of a unified mathematical language is, in fact, a problem of the design of mathematical systems based on the following major considerations:

– Optimal encoding of the basic geometric intrinsic characteristics: dimension, magnitude, direction, sense, or orientation.
– Coordinate-free methods to formulate and solve problems in physics and robotics.
– Optimal uniformity of concepts and methods across different domains, so that the intrinsic common structures are made as explicit as possible.
– Smooth articulation between different alternative systems in order to access and transfer information frictionlessly.
– Optimal computational efficiency: The computing programs using the new system should be as or more efficient than any alternative system in challenging applications.

Note that geometric algebra was constructed following these considerations, and in view of the progress of scientific theory, geometric algebra helps greatly to optimize expressions of the key ideas and consequences of the theory.

1.3 What Does Geometric Algebra Offer for Geometric Computing?

Next, we will describe the most remarkable features of geometric algebra from the perspective of geometric computing for perception–action systems.

1.3.1 Coordinate-Free Mathematical System

In geometric algebra, one writes coordinate-free expressions to capture concepts and constraints in a sort of high-level geometric reasoning approach. It is expected that the geometric information encoded in the expressions involving geometric products and the actions of operators should not suffer from interference from the reference coordinate frame. As a matter of fact, the results obtained by algebraic computing based on coordinates are geometrically meaningless or difficult to interpret geometrically. This is essentially because they are neither invariant nor covariant under the action of coordinate transformations. In fact, geometric algebra enables us to express fundamental robotics physics in a language that is free from coordinates or indices. The geometric algebra framework gives many equations a degree of clarity that is definitively lost in matrix algebra or tensor algebra.

The introduction of coordinates by R. Descartes (1596–1650) started the algebraization of the geometry. This step in geometry caused a big change from a qualitative description to a qualitative analysis. In fact, coordinates are sequences of numbers and do not have geometric meaning themselves. G. Leibniz (1646–1716) dreamed of a geometric calculus system that deals directly with geometric objects rather than with sequences of numbers. More precisely, in a mathematical system, an element of an expression should have a clear meaning of being a geometric object or a transformation operator for algebraic manipulations such as addition, subtraction, multiplication, and division.

We can illustrate the concept of independence under the action of coordinate transformations by analyzing in the Euclidean plane geometry the following operation with complex numbers $\mathbf{a}, \mathbf{b} \in \mathbb{C}$: $\bar{\mathbf{a}}\mathbf{b} := (a_1, a_2)(b_1, b_2) = (a_1b_1 - a_2b_2, a_1b_2 + a_2b_1)$. This product is not invariant under the Euclidean group, that is, the geometric information encoded in the results of the product cannot be separated from the interference of the reference coordinate frame. However, if we change the complex product to the following product: $\bar{\mathbf{a}}\mathbf{b} := (a_1, -a_2)(b_1, b_2) = (a_1b_1 + a_2b_2, a_1b_2 - a_2b_1)$, then under any rotation $r : \mathbf{a} \to \mathbf{a}e^{i\theta}$ centered at the origin, this product remains invariant: $r(\bar{\mathbf{a}})r(\mathbf{b}) := (\mathbf{a}e^{i\theta})(\mathbf{b}e^{i\theta}) = \bar{\mathbf{a}}\mathbf{b}$. Consequently, if the complex numbers are equipped with a scalar multiplication, addition, subtraction, and a geometric product, they turn from a field to a 2D geometric algebra of the 2D orthogonal geometry. It is clear that increasing the dimension of the geometric space and the generalization of the transformation group, the desired invariance will be increasingly difficult. Leibniz's dream is fulfilled for the n-D classical geometries using the framework of geometric algebras. In this book, we present the following coordinate-free geometric algebra frameworks: for 2D and 3D spaces with a Euclidean metric, for 4D spaces with a non-Euclidean metric, and for $\mathbb{R}^{n+1,1}$ spaces the conformal geometric algebras.

Assuming that we are handling expression as independently as possible upon a specific coordinate systems, in view of the implementation, one converts these expressions into low-level, coordinate-based that can be directly executed by a fast processor. In general, geometric algebra can be seen as a geometric inference engine of an automated code generator, which is able to take a high-level specification of a physical problem and automatically generate an efficient and executable implementation.

1.3.2 Models for Euclidean and Pseudo-Euclidean Geometry

When we are dealing with problems in robotics or neural computing, an important question is in which metric space we should work. In this book, we are basically concerned with three well-understood space models:

i. Models for 2D and 3D spaces with a Euclidean metric: 2D and 3D are well suited to handle the algebra of directions in the plane and 3D physical space. 3D

rotations are represented using rotors (isomorph to quaternions). You can model the kinematics of points, lines, and planes using $G_{3,0,0}$. Rotors can be used for interpolation in graphics and estimation of rotations of rigid bodies. Chapter 4 offers three increasingly powerful models for Euclidean geometry.

ii. Models for 4D spaces with non-Euclidean metric: If you are interested in linearizing a rigid motion transformation, you will need a homogeneous representation. For that, we should use a geometric algebra for the 4D space. Here, it is more convenient to choose the motor algebra $G^{+}_{3,0,1}$ described in Chap. 4. It is the algebra of Plücker lines, which can be used to model the kinematics of points, lines, and planes better than with G_3. Lines belong to the nonsingular study 6D quadric and the motors to the 8D Klein quadric. In $G^{+}_{3,0,1}$, you can formulate a motor-based equation of motion for constant velocity where in the exponent you use a bivector for twists. You can also use motors for interpolation of 3D rigid motion and estimate trajectories using EKF techniques. When you are dealing with problems of projective geometry like in computer vision, again you need a homogeneous coordinate representation, so that the image plane becomes $\mathbb{P}^2$ and the visual space $\mathbb{P}^3$. To handle the so-called n-view geometry [9] based on tensor calculus and invariant theory, you require $\mathcal{G}_{3,1}$ (Minkowski metric) for the visual space and $G_{3,0,0}$ for the image plane. This is studied in Chap. 12 [10]. Note that the intrinsic camera parameters are modeled with an affine transformation within geometric algebra as part of the projective mapping via a projective split between the projective space and the image plane. Incidence algebra, an algebra of oriented subspaces, can be used in $G_{3,1}$ and $G_{3,0,0}$ to treat problems involving geometric constraints and invariant theory.

iii. Conformal models: If you consider conformal transformations (angle preserving), conformal geometric algebra in Chap. 6 offers a non-Euclidean geometric algebra $G_{n,1}$ that includes in its multivector basis the null vectors, the origin, and point at infinity. As computational framework, it utilizes the powerful horosphere (the meet between a hyperplane and the null cone). Even though the computational framework uses a nonlinear representation for the geometric entities, one can recover the Euclidean metric. The basic geometric entity is the sphere and you can represent points, planes, lines, planes, circles, and spheres as a vectors or in dual forms, the latter being useful to reduce the complexity of algebraic expressions. As you may have noticed, the above-presented geometric algebras can be used either for kinematics in robotics or for projective geometry in computer vision. Provided that you calibrate the digital camera, you can make use of the homogeneous models from conformal geometric algebra to handle problems of robotics and those of computer vision simultaneously, however, without the need to abandon the mathematical framework. Furthermore, incidence algebra of points, lines, planes, circles, and spheres can be used in this framework as well. The topic of omnidirectional vision exploits the model of an image projected on the sphere, whereas all the problems such as rigid motion, depth, and invariant theory can also be handled using conformal geometric algebra, see Chap. 12 [10].

1.3.3 Subspaces as Computing Elements

The wedge product of k basis vectors spans a new entity, the k-vector. A set of all k-vectors spans an oriented subspace $\overset{k}{\bigwedge} V^n$. Thus, the entire geometric algebra G_n is given by

$$G_n = \overset{0}{\bigwedge} V^n \oplus \overset{1}{\bigwedge} V^n \oplus \overset{2}{\bigwedge} V^n \oplus \cdots \oplus \overset{k}{\bigwedge} V^n \oplus \cdots \oplus \overset{n}{\bigwedge} V^n.$$

Geometric algebra uses the subspace structure of the k-vector spaces to construct extended objects ranging from lines, planes, to hyperspheres. If we then represent physical objects in terms of these extended objects, we can model physical phenomena like relativistic particle motion or conformal mappings of the visual manifold into the neocortex using appropriate operators and blade transformations.

1.3.4 Representation of Orthogonal Transformations

Geometric algebra represents orthogonal transformations more efficiently than the orthogonal matrices by reducing the number of coefficients; think of the nine entries of a 3D rotation matrix and the four coefficients of a rotor. In geometric algebra, a versor product is defined as

$$\boldsymbol{O} \rightarrow \boldsymbol{V}\boldsymbol{O}\widetilde{\boldsymbol{V}},$$

where the versor acts on geometric objects of different grades, subspaces, and also on operators, quite a big difference from the matrices. The versor is applied sandwiching the object, because this is the result of successive reflections of the object with respect to hyperplanes. In geometric algebra, a physical object can be described by a $flag$, i.e., a reference using points, lines, planes, circles, and spheres crossing the gravitational center of the object. By applying versors equivalent to an orthogonal transformation on the flag, the relation of the reference geometric entities of the flag will remain invariant, i.e., topology-preserving group action.

1.3.5 Objects and Operators

A geometric object can be represented using multivector basis and wedge products, e.g., a point, line, or a plane. These geometric entities can be transformed for rigid motion, dilation, or reflection with respect to a plane or a sphere. These transformations depend on the metric of the involved space. Thus, we can model the 3D kinematics of such entities in different computational models as in the 3D Euclidean

geometric algebra $G_{3,0,0}$, 4D motor algebra $G^{+}_{3,0,1}$, or the conformal algebra $G_{4,1}$. You can see that the used transformations as versors can be applied to an object regardless the grade of its k-blades. For example, in Fig. 1.1, we can describe the arm with points, circles, and spheres and screw lines, the revoluted or prismatic joints, and then using direct/inverse kinematics relate the geometric entities from the basis through the joints until the end-effector. Here, a pertinent question will be whether or not operators can be located in space like geometric objects. The answer is yes; we can attach to any position the rotors and translators. This is a big difference from matrices; in geometric algebra, the operators or versors can be treated as geometric objects and, however, have a functional characteristic as well. In geometric algebra, objects are specified in terms of basic elements intrinsic to the problem, whereas the operators or versors are constructed depending upon which Lie group we want to use. A versor is built by successive reflections with respect to certain hyperplanes (lines, planes, and spheres). In 3D space, an example of a versor is the rotor, which is built by two successive reflections with respect to two planes that intersect the origin.

A versor is applied sandwiching a geometric object. Since a versor represents a Lie group of the general linear groups, it can also be represented as an exponential form where in the exponent the Lie algebra space is spanned with a bivector basis (Lie generators). The versor and its exponential form are quite effectively represented using bivectors, which is indeed a less redundant representation than that by the matrices. Versor-based techniques can be applied in spaces of arbitrary signature and are particularly well suited for the formulation of Lorentz and conformal transformations.

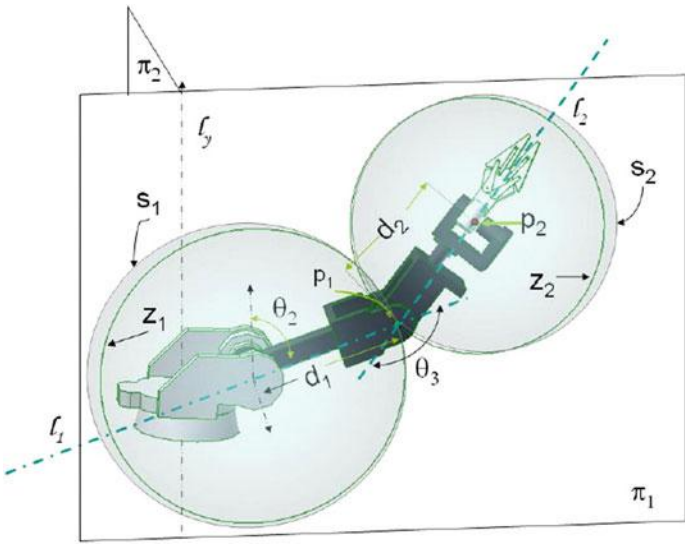


Fig. 1.1 Description of the kinematics of a 5-DOF robot arm using geometric entities of the conformal geometric algebra

In tasks of kinematics, dynamics, and modern control theory, we can exploit the Lie algebra representation acting on the bivectorial exponents rather than at the level of Lie group versor representation. In Chaps. 4 and 5, we describe bivector representations of Lie groups.

1.3.6 Extension of Linear Transformations

Linear transformations act on n-D vectors in $\mathbb{R}^n$. Since in geometric algebra a subspace is spanned by vectors, the action of the linear transformation on each individual vector will directly affect the spanned subspace. Subspaces are spanned by wedge products; an outermorphism of a subspace equals the wedge products of the transformed vectors. The outermorphism preserves the grade of any k-blade it acts on, e.g., the unit pseudoscalar must be mapped onto some multiple of itself; this multiple is the determinant of $\underline{f}(I) = \det(\underline{f})I$. We can proceed similarly when we compute the intersection of subspaces via the meet operation. We can transform linearly the result of the meet using duality and then apply an outermorphism. This would be exactly the same if we first transformed the subspaces and then computed the meet. In Chap. 12 [10], we exploit outermorphisms for computing projective invariants in the projective space and image plane.

1.3.7 Signals and Wavelets in the Geometric Algebra Framework

One may wonder why we should interest ourselves in handling n-D signals and wavelets in the geometric algebra framework. The major motivation is that since in image processing, robotics, and control engineering, n-D signals or vectors are corrupted by noise, our filters and observers should smooth signals and extract features but do so as projections on subspaces of the geometric algebra. Thinking in quadrature filters, we immediately traduce this in quaternionic filters or further we can generalize over non-Euclidean metrics the Dirac operator for multidimensional image processing. Thus, by weighting the bivectors with appropriate kernels like Gauss, Gabor, or wavelets, we can derive powerful Clifford transforms to analyze signals using the extended phase concept and carry out convolutions in a certain geometric algebra or on the Riemann sphere. So we can postulate that complex filters can be extended over bivector algebras for computing with Clifford Fourier transforms and Clifford wavelet transforms or even the space and time Fourier transform. In the geometric algebra framework, we gain a major insight and intuition for the geometric processing of noisy n-D signals. In a geometric algebra with a specific non-Euclidean metric, one can compute geometrically coupling, intrinsically different information, e.g., simultaneously process color and thermal images with a multivector derivative or regularize color optical flow with the generalized Laplacian. Chapter 11 of [10] is devoted to studying a variety of Clifford Fourier and Clifford wavelet transforms.

1.3.8 Kinematics and Dynamics

In the past, some researchers computed the direct and inverse kinematics of robot arms using matrix algebra and geometric entities like points and lines. In contrast, working in geometric algebra, the repertoire of geometric entities and the use of efficient representation of 3D rigid transformations make the computations easy and intuitive, particularly for finding geometric constraints. In conformal geometric algebra, we can perform kinematic computations using meets of spheres, planes, and screw axes, so that the resulting pair of points yields a realistic solution to the problem. Robot object manipulation together with potential fields can be reformulated in conformal geometry using a language of spheres for planning, grasping, and manipulation.

The dynamics of a robot mechanism is normally computed using a Euler Lagrange equation, where the inertial and Coriolis tensors depend on the degrees of freedom of the robot mechanism. Even though conformal geometry does not have versors for coding affine transformations, we can reformulate these equations, so that the entries of the tensors are projections of the centers of mass points of the limbs with respect to the screw axes of the joints; as a result, we can avoid quadratic entries and facilitate the estimation of the tensor parameters. This is the benefit to handling this kind of problem in either motor algebra or conformal algebra, making use of the algebra of subspaces and versors. Chapters 10 and 11 cover the computation of various problems of the kinematics and dynamics of robot mechanisms.

1.4 Solving Problems in Perception and Action Systems

In this section, we will outline how we approach the modeling and design of algorithms to handle tasks of robotic systems acting within the perception–action cycle. In the previous section, we explained what geometric algebra offers as a mathematical system for geometric computing. Here, we will be slightly more concrete and precise, illustrating the design and implementation of algorithms for real-time geometric computing.

Figure 1.2 shows an abstraction of the attitudes of many researchers and practitioners: How do they approach developing algorithms to solve problems in the domain of PAC systems? Briefly, they split the knowledge across various mathematical systems. As a consequence, as we discussed in Sect. 1.2 above, the ideas, concepts, methods, and results are unfortunately disseminated across the various mathematical systems. Being proficient in only a few of the systems, one does not have access to knowledge formulated in terms of the other mathematical systems. There is a high redundancy due to the repetitive representation of information in different mathematical systems. A deficient articulation of the different mathematical systems degrades their efficiency. The intrinsic properties and relations represented in different symbolic systems are very difficult to handle. The information density is low, because the information from the problem is reduced due to distribution over different symbolic systems.

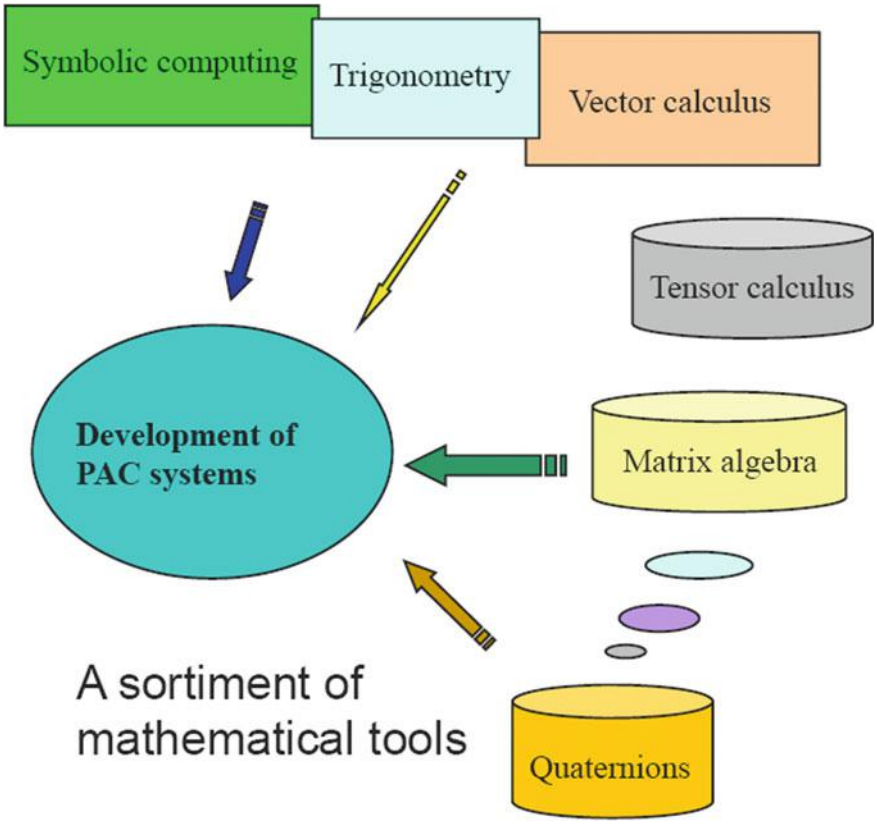


Fig. 1.2 Application of diverse mathematical systems to solve PAC problems

Bear in mind that geometric algebra was constructed for the optimal encoding of geometric intrinsic characteristics using coordinate-free methods. It ensures an optimal uniformity of concepts and methods across different domains, and it supports a smooth articulation between different alternative systems. The efficient representation of objects and operations guarantees computational efficiency. Since geometric algebra was constructed following these considerations and in view of the progress of scientific theory, geometric algebra is the adequate framework to optimize expressions of the key ideas and consequences of the theory related to perception–action systems, see Fig. 1.3.

Of course, it will not be possibly everything of the PAC problems to be formulated and computed in geometric algebra. We have to ensure that the integration of techniques proceeds in a kind of top-down approach. First, we should get acquainted with the physics of the problem aided by the contributions of researchers and paying attention particularly to how they solve the problems. Recall the old Western interpretation of the metaphor *Nanos gigantum humeris insidentes*:

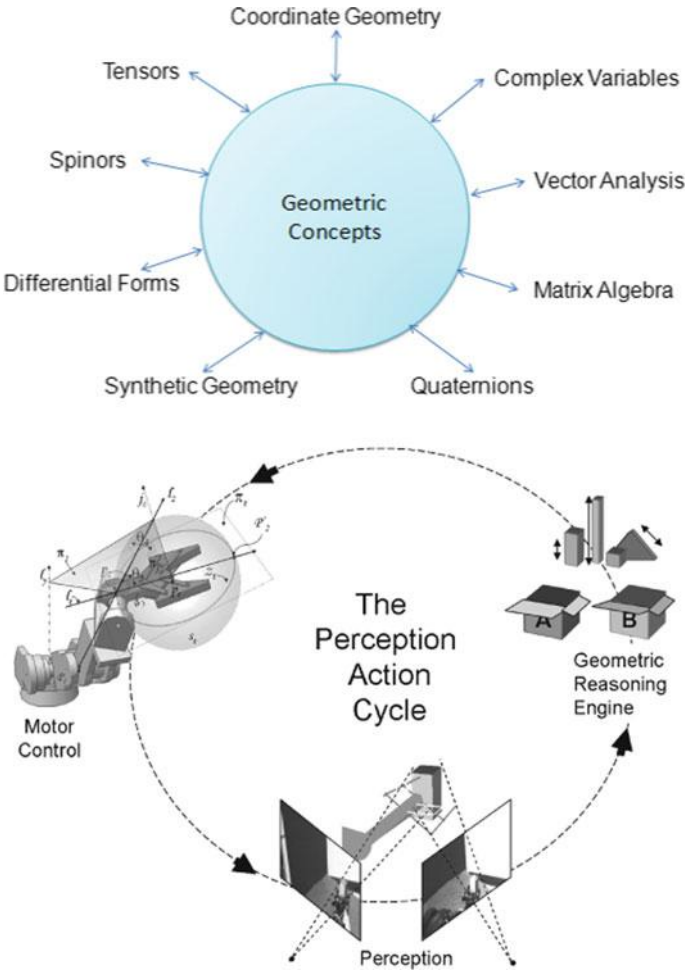


Fig. 1.3 Geometric algebra framework includes for the development of the PAC systems the essential mathematical systems

one who develops future intellectual pursuits by understanding the research and works created by notable thinkers of the past.

For a start, we should identify where we can make use of geometric algebra. Since geometric algebra is a powerful language for efficient representations and finding geometric constraints, we should first, in a high-symbolic-level approach, postulate formulas (top-down reasoning), symbolically simplify them optimally, and execute them using cost-effective and fast hardware. To close the code generation loop, the conflicts and contradictions caused by our algorithms in their application are fed back in a bottom-up fashion to a negotiation stage in order to ultimately improve our geometric algorithms.

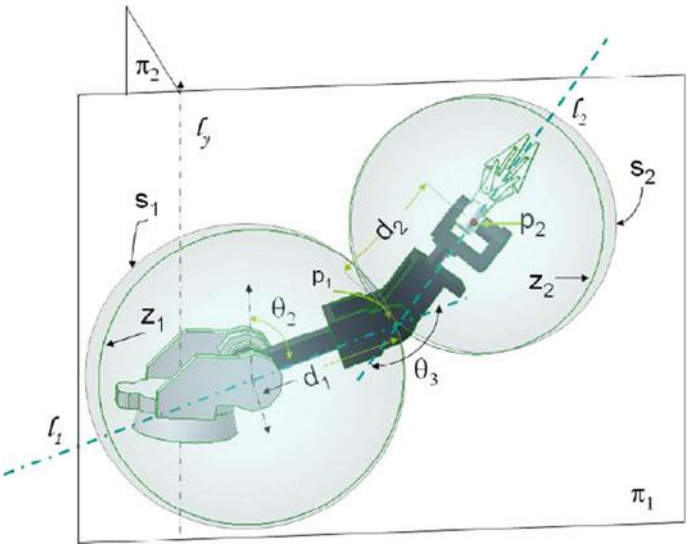


Fig. 1.4 Computing the inverse kinematics of a 5-DOF robot arm

Let us now briefly illustrate this procedure. As a first example, we compute efficiently the inverse kinematic of a robot arm. Figure 1.4 shows the rotation planes of a robot arm. While computing its inverse kinematic, one considers a circle that is the intersection of two reference spheres $z = s_1 \wedge s_2$. We then compute the pair of points ($\boldsymbol{P}\,\boldsymbol{P}$) as the meet of the swivel plane and the circle: $\boldsymbol{P}\,\boldsymbol{P} = z \wedge \pi_{swivel}$, so that we can choose one as a realistic position of the elbow. This is a point that lies on the meet of two spheres. After the optimization of the whole equation of the inverse kinematics, we get an efficient representation for computing this elbow position point and other parts of the robot arm, which can all be executed using fast hardware. Note that using a matrix formulation, it will not be possible to generate optimal code.

A second example involves the use of the meet of ruled surfaces. Imagine a laser welding hast to follow the intersection of two highly nonlinear surfaces. You can estimate these using a vision system and model them using the concept of ruled surface. In order to control the welding laser attached to the end-effector of the robot manipulator, we can follow the contour gained computing the meet of the ruled surfaces, as depicted in Fig. 1.5.

A third example is to solve the sensor-body calibration problem depicted in Fig. 1.6. This problem can be simplified and linearized, exploding the intrinsic relation of screw lines of the sensors and the robot body. The problem is reduced to finding only the unknown motors between these lines. Using matrices will yield a nonlinear problem of the kind **AX** = **XB**. As is evident, a line language of motor algebra suffices to tackle such a problem. A fourth problem entails representing 3D shapes in graphics engineering or medical image processing that traditionally should

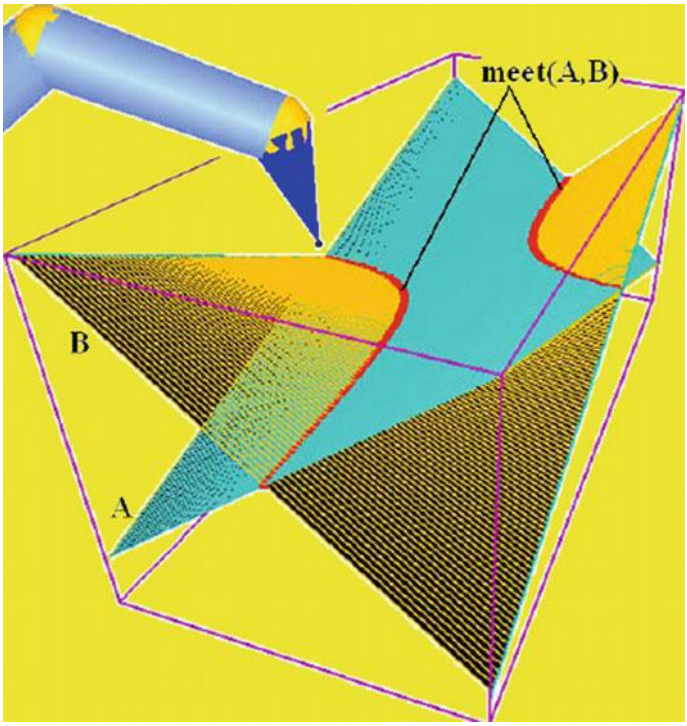


Fig. 1.5 Robot arm welding with laser along the meet of two ruled surfaces

involve the standard method called *marching cubes*. However, we generalized this method as *marching spheres* [11] using the conformal geometric algebra framework. Instead of using points on simplexes, we use spheres of $G_{4,1}$. In this way, not only do we give more expressive power to the algorithm, but we also manage to reutilize the existing software by using, instead of the 2D or 3D vectors, the 4D and 5D vectors, which represent circles and spheres in conformal geometric algebra, respectively. In Fig. 1.7, see the impressive results of carving a 3D shape, where the spheres fill the gaps better than cubes. As a fifth and last example, let us move to geometric computing. Traditionally, neural networks have been vector-based approaches with the burden of applying a coordinate-dependent algorithm for adjusting the weights between neuron layers. In this application, we have to render a contour of a certain noisy shape like a brain tumor. We use a self-organizing neural network called neural gas [12]. Instead of adjusting the weights of the neurons to locate them along the contour or shape, we adjust the exponents of motors. In this way, we are operating in the linear space of the bivector or Lie algebra. The gain is twofold: On the one hand, due to the properties of the tangential Lie algebra space, the approach is linear. On the other hand, we exploit the coordinate-free advantage by working on the Lie algebra

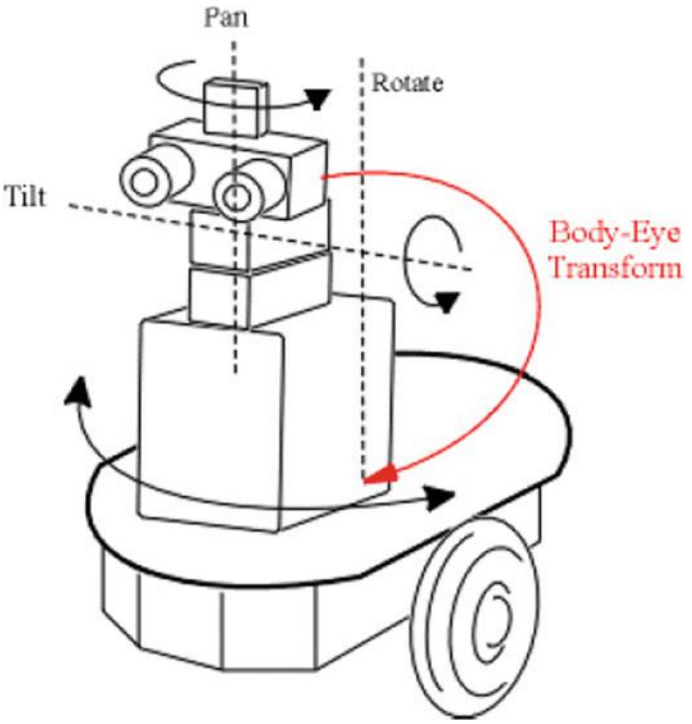


Fig. 1.6 Calibration of the coordinate system of a binocular head with respect to the robot coordinate system

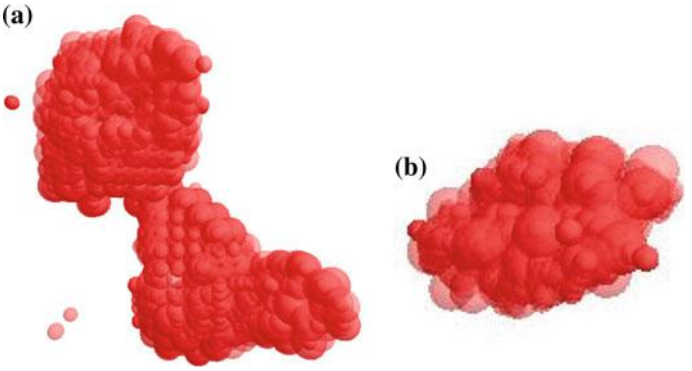


Fig. 1.7 Approximation of shape of three-dimensional objects with marching spheres; **a** approximation of the brain structure extracted from CT images (synthetic data); **b** approximation of the tumor extracted from real patient data

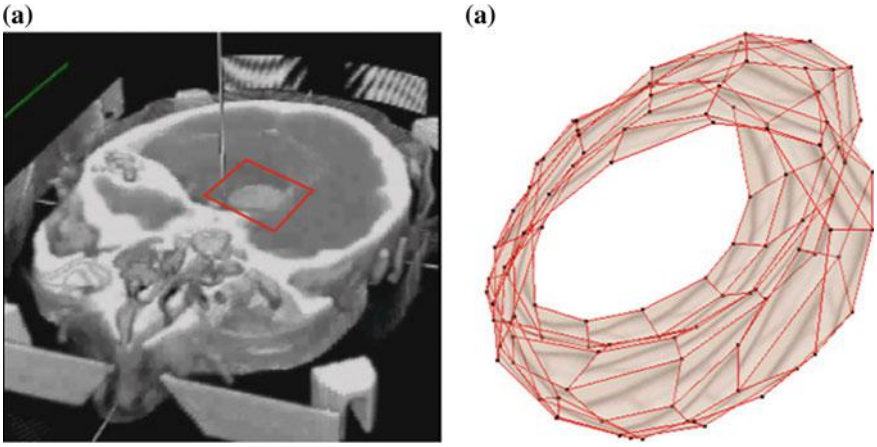


Fig. 1.8 The algorithm for 3D object's shape determination. **a** 3D model of the patient's head containing a section of the tumor in the marked region; **b** final shape after training has finished with a total of 170 versors M (associated with 170 neural units)

manifold using bivectors. Figure 1.8 shows the excellent results from segmenting a section of a brain tumor.

Finally, in the chapters of Applications I and II, the reader will find plenty of illustrations using real images and robots where we have used in a smart and creative manner the geometric algebra language. We do hope to encourage readers to use this powerful and promising framework to design new real-time algorithms for perception and action systems.

References

1. Grassmann, H. (1877). Der Ort der Hamilton'schen Quaternionen in der Ausdehnungslehre. *Mathematische Annalen*, *12*, 375.
2. Clifford, W. K. (1878). Applications of Grassmann's extensive algebra. *American Journal of Mathematics*, *1*, 350–358.
3. Hestenes, D. (1966). *Space-time algebra*. London: Gordon and Breach.
4. Hestenes, D., & Sobczyk, G. (1984). *Clifford algebra to geometric calculus: A unified language for mathematics and physics*. Dordrecht: D. Reidel.
5. Hestenes, D. (1986). *New foundations for classical mechanics*. Dordrecht: D. Reidel.
6. Hestenes, D., & Ziegler, R. (1991). Projective geometry with Clifford algebra. *Acta Applicandae Mathematicae*, *23*, 25–63.
7. Li, H., Hestenes, D., & Rockwood, A. (2001). Generalized homogeneous coordinates for computational geometry. In G. Sommer (Ed.), *Geometric computing with Clifford algebra* (pp. 27–59). New York: Springer-Verlag.
8. Hestenes, D. (2003). Orsted medal lecture 2002: Reforming the mathematical language of physics. *American Journal of Physics*, *71*(2), 104–121.
9. Hartley, R. I., & Zissermann, A. (2003). *Multiple view geometry in computer vision* (2nd ed.). Cambridge: Cambridge University Press.

10. Bayro-Corrochano, E. (2018). *Geometric algebra applications vol. I computer vision, graphics and neurocomputing*. Springer Verlag.
11. Rivera-Rovelo, J., & Bayro-Corrochano, E. (2009). The use of geometric algebra for 3D modelling and registration of medical data. *Journal of Mathematical Imaging and Vision*, *34*(1), 48–60.
12. Fritzke, B. (1995). A growing neural gas network learns topologies. In *Advances in neural information processing systems* (Vol. 7, pp. 625–632). MIT Press, Cambridge, MA.

Part I
Fundamentals of Geometric Algebra

Chapter 2
Introduction to Geometric Algebra

> . . . for geometry, you know, is the gate of science, and the gate is so low and small that one can only enter it as a little child.
> William Kingdon Clifford (1845–1879)

This chapter gives a detailed outline of geometric algebra and explains the related traditional algebras in common use by mathematicians, physicists, computer scientists, and engineers.

2.1 History of Geometric Algebra

Historically, Clifford algebra in its geometric interpretation has constituted the general framework for the embedding and development of ideas of multilinear algebra, multivariable analysis, and the representation theory of Lie groups and Lie algebras. This trend toward geometric algebra has started in 300 B.C. with the synthetic geometry of Euclid and has continued to evolve into the present. The analytic geometry of Descartes (1637), the complex algebra of Wessel and Gauss (1798), Hamilton algebra (1843), matrix algebra (Cayley, 1854), exterior algebra (Grassmann, 1844), Clifford algebra (1878), the tensor algebra of Ricci (1890), the differential forms of Cartan (1923), and the spin algebra of Pauli and Dirac (1928) have all contributed to a maturing geometric algebra framework. Geometric algebra offers a multivector concept for representation and a geometric product for multivector computation, which allow for a versatile higher order representation and computation in domains of different dimensions and metric. Complex numbers, quaternions, and dual quaternions can all be represented in both rotor and motor bivector algebras. Moreover, double, or hyperbolic, numbers can also be found in geometric algebras of positive signature. Local analysis at tangent spaces, which requires differential operations to enhance the geometric symmetries of invariants, has been done successfully using Lie algebra and Lie theory. Since the Lie algebras are isomorphic with bivector algebras, such differential operations can be advantageously implemented for

© Springer Nature Switzerland AG 2020

E. Bayro-Corrochano, *Geometric Algebra Applications Vol. II*,
https://doi.org/10.1007/978-3-030-34978-3_2

complex computations of differential geometry, as in the recognition of higher order symmetries. Projective geometry and multilinear algebra, too, are elegantly reconciled in Clifford algebra, providing the resulting algebra of incidence with the duality principle, inner product, and outermorphisms.

In the middle of the nineteenth century, J. Liouville proved, for the three-dimensional case, that any conformal mapping on the whole of $\mathbb{R}^n$ can be expressed as a composition of *inversions in spheres* and *reflections in hyperplanes* [1]. In particular, *rotation*, *translation*, *dilation*, and *inversion* mappings will be obtained with these two mappings. In conformal geometric algebra, these concepts are simplified, because due to the isomorphism between the conformal group on $\mathbb{R}^n$ and the Lorentz group on $\mathbb{R}^{n+1}$, we can easily express, with a linear Lorentz transformation, a nonlinear conformal transformation, and then use *versor* representation to simplify *composition of transformations* with *multiplication of vectors* [2]. Thus, with conformal geometric algebra, it is computationally more efficient and simpler to interpret the geometry of the conformal mappings than with matrix algebra. Conformal geometric algebra is the fusion of the Clifford geometric algebra and the non-Euclidean hyperbolic geometry. One of the results of the non-Euclidean geometry demonstrated by Nikolai Lobachevsky in the nineteenth century is that in spaces with hyperbolic structure we can find subsets that are isomorphic to a Euclidean space. In order to do this, Lobachevsky introduced two constraints, to what is now called the *conformal point* $\boldsymbol{x}_c \in \mathbb{R}^{n+1,1}$. The first constraint is the *homogeneous* representation, *normalizing* the vector $\boldsymbol{x}_c$ such that $\boldsymbol{x}_c \cdot e_\infty = -1$, and the second constraint is such that the vector must be a *null vector*, that is, $\boldsymbol{x}_c^2 = 0$.

Conformal geometric algebra offers enormous advantages by computing in the null cone space. The space $\mathbb{R}^n$ is extended with two null vectors (the origin and the point at infinity), which are used to represent the basic computational unit, the sphere. In conformal geometric algebra, one represents points, lines, planes, and volumes using unit spheres. Conformal transformations are represented using combinations of versors that are generated via reflections with respect to spheres. Its homogeneous representation allows projective computations; via the inner product, one recovers the Euclidean metric. As a result, conformal geometric algebra constitutes a suitable computational framework for dealing with a variety of conformal and projective problems involving time dependence as in causality and space and time analysis. One of the criticisms of conformal geometric algebra is the wrong idea that this system can manipulate only basic entities (points, lines, planes, and spheres), and therefore it won't be useful to model general two- and three-dimensional objects, curves, surfaces, or any other nonlinear entity required to solve a problem of a perception–action system in robotics and computer vision. Surprisingly, ruled surfaces can be treated in conformal geometric algebra for promising applications like the motion guidance of very nonlinear curves, reaching and 3D object manipulation on very nonlinear surfaces.

Our initial attempts to use geometric algebra in have been successful, reinforcing our opinion that there is no need to abandon this framework in order to carry out different kinds of computations. For all of these reasons, we believe that the single unifying language of geometric algebra offers the strongest potential for buil-

ding perception and action systems; it allows us better to understand links between different fields, incorporate techniques from one field into another, reformulate old procedures, and find extensions by widening their sphere of applicability. Finally, geometric algebra helps to reduce the complexity of algebraic expressions, and as a result improves algorithms both in speed and accuracy.

2.2 What Is Geometric Algebra?

The algebras of Clifford and Grassmann are well known to pure mathematicians, but in the beginning they were abandoned by physicists in favor of the vector algebra of Gibbs, the commonly used algebra today in most areas of physics. Geometric algebra is a coordinate-free approach to geometry based on the algebras of Grassmann [3] and Clifford [4]. The geometric approach to Clifford algebra adopted in this book was pioneered in the 1960s by Hestenes [5], who has since worked on developing his version of Clifford algebra—which will be referred to as *geometric algebra* in this volume—into a unifying language for mathematics and physics [6, 7]. Hestenes also presented a study of projective geometry using Clifford algebra [8] and recently the essential concepts of the conformal geometric algebra [2]. The introductory sections of this chapter will present the basic definitions of geometric algebra.

The reader can find, in Appendix A (Chap. 20 [9]), an outline of classical Clifford algebras with explanations of its related subjects and many complementary mathematical definitions which will help the more mathematically inclined reader to have a broad view of the role of geometric algebra and Clifford algebra. Also, a reader with no mathematical background can consult this Appendix to learn more about some concepts and related definitions.

For this mathematical system, we denote scalars with lowercase letters, matrices with uppercase letters, and we use bold lowercase for both vectors in three dimensions and the bivector parts of spinors. Spinors and dual quaternions in four dimensions are denoted by bold uppercase letters.

2.2.1 Basic Definitions

Let V^n be a vector space of dimension n. We are going to define and generate an algebra G_n, called a *geometric algebra*. Let $\{e_1, e_2, \ldots, e_n\}$ be a set of orthonormal basis vectors of V^n. The geometric algebra G_n is equipped with a quadratic form Q. The geometric algebra $G(V^n, Q)$ is generated by V^n subject to the condition

$$v^2 = Q(v), \quad \forall v \in V^n. \tag{2.1}$$

This is the fundamental Clifford identity. If the characteristic of the ground field F is not 2, this condition can be rewritten in the following form:

$$uv + vu = 2\langle u, v\rangle, \quad \forall u, v \in V^n, \tag{2.2}$$

where $\langle u, v\rangle = \frac{1}{2}(Q(u+v) - Q(u) - Q(v))$ is the symmetric bilinear form associated to Q. Regarding the quadratic form Q, one can notice that Clifford or geometric algebras are closely related to exterior algebras. As a matter of fact, if $Q = 0$ then the Clifford or geometric algebra $G(V^n, Q)$ is just the exterior algebra $\bigwedge(V^n)$. Since the Clifford or geometric product includes the extra information of the quadratic form Q, the Clifford or geometric product is fundamentally richer than the exterior product. According to Eq. (2.2), the product of the orthonormal basis vectors is anticommutative, namely,

$$e_j e_k + e_k e_j = 2\langle e_j, e_k\rangle = 0 \rightarrow e_j e_k = -e_k e_j, \quad \forall j \neq k. \tag{2.3}$$

The *scalar multiplication* and *sum* in G_n are defined in the usual way of a vector space. The *product* or *geometric product* of elements of the basis of G_n will simply be denoted by juxtaposition. In this way, from any two basis vectors e_j and e_k, a new element of the algebra is obtained, and denoted as $e_j e_k \equiv e_{jk}$.

The basis vectors must *square* in $+1$, -1, or 0; this means that there are nonnegative integers p, q, and r such that $n = p + q + r$ and

$$e_i e_i = e_i^2 = \begin{cases} +1 \text{ for } i = 1, \ldots, p, \\ -1 \text{ for } i = p+1, \ldots, p+q, \\ \ \ 0 \text{ for } i = p+q+1, \ldots, n. \end{cases} \tag{2.4}$$

This *product* will be called the *geometric product* of G_n. With these operations, G_n is an associative linear algebra with identity and is called the *geometric algebra* or *Clifford algebra* of dimension $n = p + q + r$, generated by the vector space V^n. It is usual to write G_n for the geometric algebra spanned by the real vector space V^n. The elements of this geometric algebra are called *multivectors*, because they are entities generated by the sum of elements of *mixed grade* of the basis set of G_n, such as

$$\mathbf{A} = \langle \mathbf{A}\rangle_0 + \langle \mathbf{A}\rangle_1 + \cdots + \langle \mathbf{A}\rangle_n, \tag{2.5}$$

where the multivector $\mathbf{A} \in G_n$ is expressed by the addition of its 0-*vector* part (scalar) $\langle \mathbf{A}\rangle_0$, its 1-*vector* part (vector) $\langle \mathbf{A}\rangle_1$, its 2-*vector* part (bivector) $\langle \mathbf{A}\rangle_2$, its 3-*vector* part (trivector) $\langle \mathbf{A}\rangle_3$, …, and its n-*vector* part $\langle \mathbf{A}\rangle_n$. We call an r-*blade* or a *blade of grade* r, to the geometric product of r linearly independent vectors. Because the addition of k-vectors (homogeneous vectors of grade k) is closed and the multiplication of a k-vector by a scalar is another k-vector, the set of all k-vectors is a vector space, denoted $\bigwedge^k V^n$. Each of these spaces is spanned by $\binom{n}{k}$ k-vectors, where $\binom{n}{k} = \frac{n!}{(n-k)!k!}$. Thus, our geometric algebra G_n, which is spanned by $\sum_{k=0}^{n} \binom{n}{k} = 2^n$ elements, is a direct sum of its homogeneous subspaces of grades 0, 1, 2, …, n, that is,

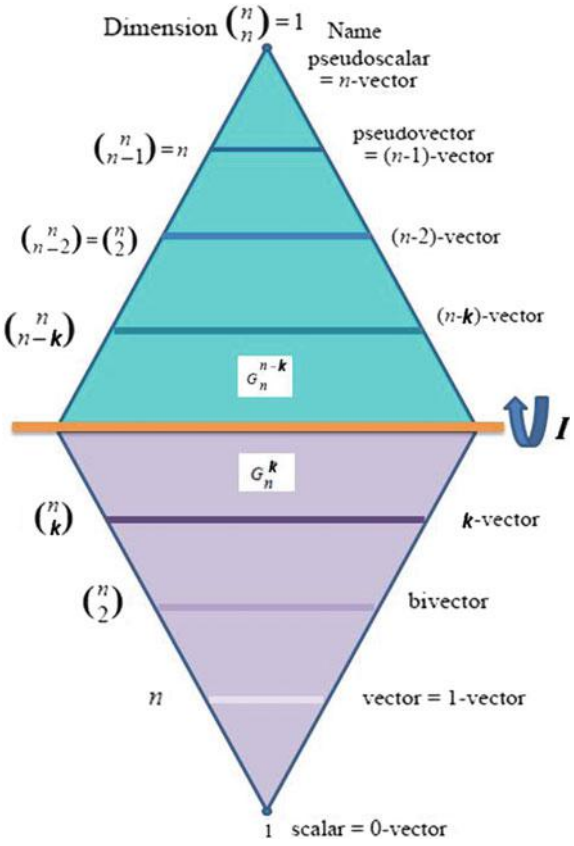


Fig. 2.1 Schematic representation of G_n

$$G_n = \bigwedge^0 V^n \oplus \bigwedge^1 V^n \oplus \bigwedge^2 V^n \oplus \cdots \oplus \bigwedge^k V^n \oplus \cdots \oplus \bigwedge^n V^n, \qquad (2.6)$$

where $\bigwedge^0 V^n = R$ is the set of real numbers and $\bigwedge^1 V^n = V^n$. Thus, any multivector of G_n can be expressed in terms of the basis of these subspaces. The biggest grade k-blade or hypervolume is called the pseudoscalar, and it is computed as the geometric product of all vector bases, i.e., $I = e_1 e_e \ldots e_n$.

Figure 2.1 depicts a schematic representation of G_n. It shows the duality symmetry of G_n. The lower triangle corresponds to the k-vectors, and it is reflected to the upper triangle of the $(n-k)$-vectors; note that the factorial number of k-vectors corresponds to those of the Pascal triangle. This duality symmetry is generated via the dual operator or pseudoscalar I, so the dual of the k-vector is the $(n-k)$-vector, i.e., $(n-k)$-vector = $I(k$-vector). From the bottom-up, the r-th rung of the ladder represents the space of k-vectors $\binom{n}{k}$. The length of the rung corresponds to the dimension of the space, which increases from the bottom-up and the top-down to the

middle. The one-dimensional space of scalars is represented by a single point. Via the dual operator or pseudoscalar I, a duality transformation of the elements of G_n in the lower triangle is flip about its horizontal symmetry axis.

For instance, in the geometric algebra G_3 with multivector basis

$$\{1, e_1, e_2, e_3, e_{12}, e_{31}, e_{23}, I_3 = e_{123}\}, \tag{2.7}$$

in the lower triangle there are the scalar, three vectors and in the upper triangle their duals, three bivectors and the pseudoscalar.

A *typical* multivector u will be of the form

$$\mathbf{u} = \alpha_0 + \alpha_1 e_1 + \alpha_2 e_2 + \alpha_3 e_3 + \alpha_4 e_{12} + \alpha_5 e_{31} + \alpha_6 e_{23} + \alpha_7 I_3, \tag{2.8}$$

where the α_i's are real numbers. When a subalgebra is generated only by multivectors of even grade, it is called the *even* subalgebra of G_n and will be denoted as G_n^+. Note, however, that the set of odd multivectors is not a subalgebra of G_n.

In $G_{p,q,r}$, we have the *pseudoscalar* $I_n \equiv e_{12\cdots n}$, but we can even define a *pseudoscalar* for each set of vectors that square in a positive number, negative number, and zero, that is, $I_p \equiv e_{12\cdots p}$, $I_q \equiv e_{(p+1)(p+2)\cdots(p+q)}$, and $I_r \equiv e_{(p+q+1)(p+q+2)\cdots(p+q+r)}$, where $n = p + q + r$. We say that $G_{p,q,r}$ is degenerated if $r \neq 0$.

2.2.2 Nonorthonormal Frames and Reciprocal Frames

A frame $\{e_1, e_2, \ldots, e_n\}$ span a n-dimensional vector space $\mathbb{R}^n$, and it can be orthonormal or not orthonormal. Consider a set of n linearly independent vectors $\{e_k\}$ which are not orthonormal. Any vector $\boldsymbol{v}$ can be represented uniquely in tensor notation using this set

$$\boldsymbol{v} = v^k e_k. \tag{2.9}$$

To find the components (v^k), we resort to the use of the so-called *reciprocal frame* which fulfills

$$e^i \cdot e_j = \delta^i_j. \tag{2.10}$$

By computing the inner product of $\{e_k\}$ with the vector $\boldsymbol{v}$, one gets the (v^k)

$$e^k \cdot \boldsymbol{v} = e^k \cdot (v^k e_k) = e^i \cdot e_j v^j = v^j \delta^i_j = v^k. \tag{2.11}$$

In order to construct a reciprocal frame, we can use an entirely geometric approach. We begin with the first e^1 in which it has to lie outside of the hyperplane $\Pi = e_2 \wedge e_3 \wedge \cdots \wedge e_n\}$. The vector perpendicular to Π is computed via dualization. For

that, we use the dualization operator or pseudoscalar I

$$e^1 = \alpha e_2 \wedge e_3 \wedge \cdots \wedge e_n I, \tag{2.12}$$

where alpha is some constant, which can be fixed by applying the inner product with e^1

$$\begin{aligned} 1 = e_1 \cdot e^1 &= \alpha e_1 \cdot (e_2 \wedge e_3 \wedge \cdots \wedge e_n I) \\ &= \alpha e_1 \wedge e_2 \wedge e_3 \wedge \cdots \wedge e_n I. \end{aligned} \tag{2.13}$$

Substituting the volume element

$$\boldsymbol{V}_n = e_1 \wedge e_2 \cdots \wedge e_n \tag{2.14}$$

in previous equation, one gets

$$1 = \alpha e_1 \wedge e_2 \cdots \wedge e_n I = \alpha \boldsymbol{V}_n I, \tag{2.15}$$

thus $\alpha = I^{-1}\boldsymbol{V}_n - 1$ which substituted in Eq. (2.12) yields

$$e^1 = e_2 \wedge e_3 \cdots \wedge e_n \boldsymbol{V}_{-1}. \tag{2.16}$$

This result can be extended straightforwardly to compute the other e^k of the reciprocal frame

$$e^k = (-1)^{k+1} e_1 \cdots \wedge \check{e}_2 \wedge \cdots \wedge e_n \boldsymbol{V}_n^{-1}, \tag{2.17}$$

where the check indicates that the term e_k is left out in the wedge product.

2.2.3 Reciprocal Frames with Curvilinear Coordinates

In many situations, one works in non-Cartesian coordinate systems, where a coordinate system is defined by a set of scalar function $\{x^i(\boldsymbol{x})\}$ that is defined over some region. A function $F(\boldsymbol{x})$ can be expressed in terms of such coordinates as $F(x^i)$. If we apply the chain rule,

$$\nabla F = \nabla x^i \partial_i F = e^i \partial_i F, \tag{2.18}$$

this defines the so-called contravariant frame vectors $\{e^i\}$ as follows:

$$e^i = \nabla x^i. \tag{2.19}$$

In Euclidean space, these vectors are fully perpendicular to the surfaces of constant x^i. It is clear that these vectors have vanishing curl:

$$\nabla \wedge \boldsymbol{e}^i = \nabla \wedge (\nabla x^i) = 0. \tag{2.20}$$

The reciprocal frame vectors are the coordinate vectors, which are covariant:

$$e_i = \partial_i x, \tag{2.21}$$

which are formed incrementing the x^i coordinates by keeping all others fixed. These two frames are reciprocal, because they fulfill

$$e_i \cdot e^j = (\partial_i x) \cdot \nabla x^j = \partial_i x^j = \delta_j^i. \tag{2.22}$$

Particularly for the case when the space signature is not Euclidean, we should refrain from using these kinds of frame representations restricted to orthogonal frames and compensated with weighting factors as follows:

$$e_i = g_i \hat{e}_i, \qquad e^i = g_i^{-1} \hat{e}_i. \tag{2.23}$$

2.2.4 Some Useful Formulas

In [10], introduce the useful result which we will present next and use to recover a rotor in Sect. 4.3.3. Using the basic identity

$$\boldsymbol{x} = \boldsymbol{x}^k e_k = \boldsymbol{x} \cdot e^k e_k = \boldsymbol{x} \cdot e_k e^k, \tag{2.24}$$

many useful results can be derived; consider first

$$e_k e^k \cdot (\boldsymbol{x} \wedge \boldsymbol{y}) = e_k (e^k \cdot \boldsymbol{x}\boldsymbol{y} - e^k \cdot \boldsymbol{y}\boldsymbol{x}) = \boldsymbol{x}\boldsymbol{y} - \boldsymbol{y}\boldsymbol{x} = 2\boldsymbol{x} \wedge \boldsymbol{y}. \tag{2.25}$$

Using induction and utilizing again Eq. (2.24), this result can be generalized for a p-grade multivector $\boldsymbol{X}_p$

$$e_k e^k \cdot \boldsymbol{X}_p = p\boldsymbol{X}_p. \tag{2.26}$$

Consider now

$$e_k e^k = e_k (e_j \cdot e_k e^j) = e_j \cdot e_k e^k e^j. \tag{2.27}$$

Note that the inner product $e_j \cdot e_k$ is symmetric with respect to j, k; thus one can only pick up the component of $e^k e^j$. This is only a scalar; thus one can only get a scalar contribution to the sum, namely,

$$e_k e^k = e_k \cdot e^k = n, \tag{2.28}$$

where n corresponds to the dimension of the space. Now computing the wedge product

$$e_k e^k \wedge \boldsymbol{X}_p = e_k(e^k \boldsymbol{X}_p - e^k \cdot \boldsymbol{X}_p) = (n-p)\boldsymbol{X}_p, \tag{2.29}$$

and combining the above results, one gets finally

$$e_k \boldsymbol{X}_p e^k = (-1)^p e_k(e^k \wedge \boldsymbol{X}_p - e^k \cdot \boldsymbol{X}_p) = (-1)^p (n-2p)\boldsymbol{X}_p. \tag{2.30}$$

2.3 Multivector Products

It will be convenient to define other products between the elements of this algebra, which will allow us to set up several geometric relations (unions, intersections, projections, etc.) between different geometric entities (points, lines, planes, spheres, etc.) in a very simple way.

First, we define the *inner product*, $\boldsymbol{a} \cdot \boldsymbol{b}$, and the *exterior or wedge product*, $\boldsymbol{a} \wedge \boldsymbol{b}$, of any two 1-*vectors* $\boldsymbol{a}, \boldsymbol{b} \in G_n$, as the *symmetric* and *antisymmetric* parts of the geometric product $\boldsymbol{ab}$, respectively. That is, from the expression

$$\boldsymbol{ab} = \overbrace{\frac{1}{2}(\boldsymbol{ab} + \boldsymbol{ba})}^{\text{symmetric part}} + \overbrace{\frac{1}{2}(\boldsymbol{ab} - \boldsymbol{ba})}^{\text{antisymmetric part}}, \tag{2.31}$$

we define the *inner product*

$$\boldsymbol{a} \cdot \boldsymbol{b} \equiv \frac{1}{2}(\boldsymbol{ab} + \boldsymbol{ba}) \tag{2.32}$$

and the *outer* or *wedge product*

$$\boldsymbol{a} \wedge \boldsymbol{b} \equiv \frac{1}{2}(\boldsymbol{ab} - \boldsymbol{ba}). \tag{2.33}$$

Thus, we can now express the geometric product of two vectors in terms of these two new operations as

$$\boldsymbol{ab} = \boldsymbol{a} \cdot \boldsymbol{b} + \boldsymbol{a} \wedge \boldsymbol{b}. \tag{2.34}$$

From (2.32) and (2.33), $\boldsymbol{a} \cdot \boldsymbol{b} = \boldsymbol{b} \cdot \boldsymbol{a}$ and $\boldsymbol{a} \wedge \boldsymbol{b} = -\boldsymbol{b} \wedge \boldsymbol{a}$.

The inner product of the vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ corresponds to the standard *scalar* or *dot* product and produces a scalar. In a subsection ahead, we will discuss in detail other complementary definition of inner product called *contraction* introduced by Lounesto [11]. The outer or wedge product of these vectors is a new quantity,

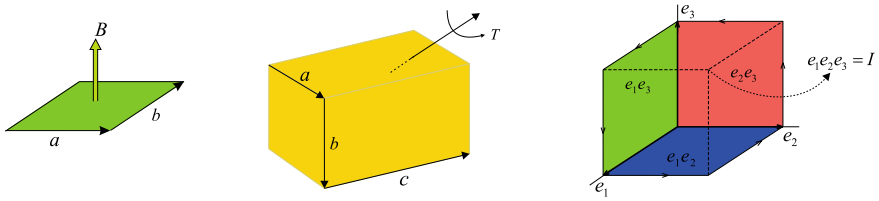


Fig. 2.2 **a** (left) The directed area, or bivector, $\boldsymbol{B} = \boldsymbol{a} \wedge \boldsymbol{b}$; **b** (middle) the oriented volume, or trivector, $\boldsymbol{T} = \boldsymbol{a} \wedge \boldsymbol{b} \wedge \boldsymbol{c}$; **c** (right) multivector basis of the Euclidean 3D space

which we call a *bivector*. We can think of a bivector as an oriented area in the plane containing $\boldsymbol{a}$ and $\boldsymbol{b}$ formed by sweeping $\boldsymbol{a}$ along $\boldsymbol{b}$, see Fig. 2.2a. Thus, $\boldsymbol{b} \wedge \boldsymbol{a}$ will have an opposite orientation, because the wedge product is anticommutative. The outer product is immediately generalizable to higher dimensions: for example, $(\boldsymbol{a} \wedge \boldsymbol{b}) \wedge \boldsymbol{c}$, a *trivector*, is interpreted as the oriented volume formed by sweeping the area $\boldsymbol{a} \wedge \boldsymbol{b}$ along vector $\boldsymbol{c}$, see Fig. 2.2b. Note that the outer product does not determine the spatial forms of the bivector and trivector: it computes only their orientations and values of their involved determinants corresponding to the area and volume, respectively. Figure 2.2a, b depicts such forms simply to favor the geometric understanding. Figure 2.2c depicts the multivector basis of the Euclidean 3D space.

The symmetric part of the geometric product of two vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ is called the *anticommutator product* and the antisymmetric part the *commutator product*. These expressions are denoted as follows:

$$\boldsymbol{a} \overline{\times} \boldsymbol{b} = \frac{1}{2}(\boldsymbol{ab} + \boldsymbol{ba}) \qquad \texttt{anticommutator product}, \tag{2.35}$$

$$\boldsymbol{a} \underline{\times} \boldsymbol{b} = \frac{1}{2}(\boldsymbol{ab} - \boldsymbol{ba}) \qquad \texttt{commutator product}. \tag{2.36}$$

In general, the geometric product of two multivectors $\boldsymbol{A}, \boldsymbol{B} \in G_n$ can be written as the sum of their commutator and anticommutator:

$$\boldsymbol{AB} = \frac{1}{2}(\boldsymbol{AB} + \boldsymbol{BA}) + \frac{1}{2}(\boldsymbol{AB} - \boldsymbol{BA}) = \boldsymbol{A} \overline{\times} \boldsymbol{B} + \boldsymbol{A} \underline{\times} \boldsymbol{B}. \tag{2.37}$$

In the literature, the commutator of two multivectors $\boldsymbol{A}, \boldsymbol{B} \in G_n$ is usually written as $[\boldsymbol{A}, \boldsymbol{B}]$ and the anticommutator as $\{\boldsymbol{A}, \boldsymbol{B}\}$, although that we prefer to emphasize their role within the geometric product by using the operator symbols $\overline{\times}$ and $\underline{\times}$.

The geometric product of a vector and a bivector reads

$$\begin{aligned} \boldsymbol{a}(\boldsymbol{b} \wedge \boldsymbol{c}) &= \frac{1}{2}\boldsymbol{a}(\boldsymbol{bc} - \boldsymbol{cb}) \\ &= (\boldsymbol{a} \cdot \boldsymbol{b})\boldsymbol{c} - (\boldsymbol{a} \cdot \boldsymbol{c})\boldsymbol{b} - \frac{1}{2}(\boldsymbol{bac} - \boldsymbol{cab}) \end{aligned}$$

$$\begin{aligned} &= 2(\boldsymbol{a}\cdot\boldsymbol{b})\boldsymbol{c} - 2(\boldsymbol{a}\cdot\boldsymbol{c})\boldsymbol{b} + \frac{1}{2}(\boldsymbol{bc}-\boldsymbol{ca})\boldsymbol{a} \\ &= 2(\boldsymbol{a}\cdot\boldsymbol{b})\boldsymbol{c} - 2(\boldsymbol{a}\cdot\boldsymbol{c})\boldsymbol{b} + (\boldsymbol{a}\wedge\boldsymbol{c})\boldsymbol{a}. \end{aligned} \tag{2.38}$$

From this result, one gets an extension of the definition of the inner product:

$$\boldsymbol{a}\cdot(\boldsymbol{b}\wedge\boldsymbol{c}) = \frac{1}{2}[\boldsymbol{a}(\boldsymbol{b}\wedge\boldsymbol{c}) - (\boldsymbol{b}\wedge\boldsymbol{c})\boldsymbol{a}] = (\boldsymbol{a}\cdot\boldsymbol{b})\boldsymbol{c} - (\boldsymbol{a}\cdot\boldsymbol{c})\boldsymbol{b}. \tag{2.39}$$

The remaining part of the product corresponds to a trivector:

$$\boldsymbol{a}\wedge(\boldsymbol{b}\wedge\boldsymbol{c}) = \frac{1}{2}[\boldsymbol{a}(\boldsymbol{b}\wedge\boldsymbol{c}) + (\boldsymbol{b}\wedge\boldsymbol{c})\boldsymbol{a}] = \boldsymbol{a}\wedge\boldsymbol{b}\wedge\boldsymbol{c}. \tag{2.40}$$

So the geometric product of a vector and a bivector is

$$\boldsymbol{a}(\boldsymbol{b}\wedge\boldsymbol{c}) = \boldsymbol{a}\cdot(\boldsymbol{b}\wedge\boldsymbol{c}) + \boldsymbol{a}\wedge(\boldsymbol{b}\wedge\boldsymbol{c}) = (\boldsymbol{a}\cdot\boldsymbol{b})\boldsymbol{c} - (\boldsymbol{a}\cdot\boldsymbol{c})\boldsymbol{b} + \boldsymbol{a}\wedge\boldsymbol{b}\wedge\boldsymbol{c}. \tag{2.41}$$

The reader can see that in geometric algebra one has to handle expressions with large number of brackets; thus, we should follow an operator ordering convention: in the absence of brackets, inner and outer products take precedence over the geometric products, so we can write

$$(\boldsymbol{a}\cdot\boldsymbol{b})\boldsymbol{c} = \boldsymbol{a}\cdot\boldsymbol{bc}, \tag{2.42}$$

where the right-hand side is not to be confused with $\boldsymbol{a}\cdot(\boldsymbol{bc})$.

2.3.1 Further Properties of the Geometric Product

The geometric product of a vector and a bivector extends simply to that of a vector $\boldsymbol{a}$ and a grade-r multivector $\boldsymbol{A}_r$ (composed of r orthogonal blades $\boldsymbol{a}_r$). To carry out the geometric product, first decompose $\boldsymbol{A}_r$ into blades

$$\begin{aligned} \boldsymbol{aA}_r &= (\boldsymbol{aa}_1)\boldsymbol{a}_2\ldots\boldsymbol{a}_r = (\boldsymbol{a}\cdot\boldsymbol{a}_1)\boldsymbol{a}_2\ldots\boldsymbol{a}_r + (\boldsymbol{a}\wedge\boldsymbol{a}_1)\boldsymbol{a}_2\ldots\boldsymbol{a}_r \\ &= 2(\boldsymbol{a}\cdot\boldsymbol{a}_1)\boldsymbol{a}_2\ldots\boldsymbol{a}_r - (\boldsymbol{a}_1\boldsymbol{a})\boldsymbol{a}_2\ldots\boldsymbol{a}_r \\ &= 2\boldsymbol{a}\cdot\boldsymbol{a}_1\boldsymbol{a}_2\ldots\boldsymbol{a}_r - 2\boldsymbol{a}\cdot\boldsymbol{a}_2\boldsymbol{a}_1\boldsymbol{a}_3\ldots\boldsymbol{a}_r + \boldsymbol{a}_1\boldsymbol{a}_2\boldsymbol{a}\boldsymbol{a}_3\ldots\boldsymbol{a}_r \\ &= 2\sum_{k=1}^{r}(-1)^{k+1}\boldsymbol{a}\cdot\boldsymbol{a}_k\boldsymbol{a}_1\boldsymbol{a}_2\ldots\check{\boldsymbol{a}}_k\ldots\boldsymbol{a}_r + (-1)^r\boldsymbol{a}_1\boldsymbol{a}_2\ldots\boldsymbol{a}_r\boldsymbol{a}, \end{aligned} \tag{2.43}$$

where the check indicates that this term is missing from the series. Each term in the sum has grade $r-1$, so one can define

$$\boldsymbol{a} \cdot \boldsymbol{A}_r = \langle \boldsymbol{a}\boldsymbol{A}_r \rangle_{r-1} = \frac{1}{2}(\boldsymbol{a}\boldsymbol{A}_r - (-1)^r \boldsymbol{A}_r \boldsymbol{a}) \tag{2.44}$$

and the right term is the remaining antisymmetric part. One can therefore write

$$\boldsymbol{a} \wedge \boldsymbol{A}_r = \langle \boldsymbol{a}\boldsymbol{A}_r \rangle_{r+1} = \frac{1}{2}(\boldsymbol{a}\boldsymbol{A}_r + (-1)^r \boldsymbol{A}_r \boldsymbol{a}). \tag{2.45}$$

Thus, the geometric product of a vector and a grade-r multivector $\boldsymbol{A}_r$ can be written as

$$\boldsymbol{a}\boldsymbol{A}_r = \boldsymbol{a} \cdot \boldsymbol{A}_r + \boldsymbol{a} \wedge \boldsymbol{A}_r = \langle \boldsymbol{a}\boldsymbol{A}_r \rangle_{r-1} + \langle \boldsymbol{a}\boldsymbol{A}_r \rangle_{r+1}. \tag{2.46}$$

Geometric product multiplication by a vector lowers and raises the grade of a multivector by 1. In Eq. (2.43), we assumed that the vectors are orthogonal: we can extend this decomposition for the case of nonorthogonal vectors as follows:

$$\begin{aligned} \boldsymbol{a} \cdot (\boldsymbol{a}_1 \wedge \boldsymbol{a}_2 \wedge \ldots \boldsymbol{a}_r) &= \frac{1}{2}[\boldsymbol{a}\langle \boldsymbol{a}_1 \boldsymbol{a}_2 \ldots \boldsymbol{a}_r \rangle_r - (-1)^r \langle \boldsymbol{a}_1 \boldsymbol{a}_2 \wedge \ldots \boldsymbol{a}_r \rangle_r \boldsymbol{a}] \\ &= \frac{1}{2}\langle \boldsymbol{a}\boldsymbol{a}_1 \boldsymbol{a}_2 \wedge \ldots \boldsymbol{a}_r - (-1)^r \boldsymbol{a}_1 \boldsymbol{a}_2 \wedge \ldots \boldsymbol{a}_r \boldsymbol{a} \rangle_{r-1}. \end{aligned} \tag{2.47}$$

In this equation, since we are expecting to lower the r-grade by 1, we can insert $\boldsymbol{a}$ within the brackets. Now, by reutilizing Eq. (2.43), we can write

$$\begin{aligned} \boldsymbol{a} \cdot (\boldsymbol{a}_1 \wedge \boldsymbol{a}_2 \wedge \cdots \wedge \boldsymbol{a}_r) &= \left\langle \sum_{k=1}^{r} (-1)^{k+1} \boldsymbol{a} \cdot \boldsymbol{a}_k \boldsymbol{a}_1 \boldsymbol{a}_2 \ldots \check{\boldsymbol{a}}_k \ldots \boldsymbol{a}_r \right\rangle_{r-1} \\ &= \sum_{k=1}^{r} (-1)^{k+1} \boldsymbol{a} \cdot \boldsymbol{a}_k \boldsymbol{a}_1 \boldsymbol{a}_2 \ldots \check{\boldsymbol{a}}_k \ldots \boldsymbol{a}_r. \end{aligned} \tag{2.48}$$

This result is very useful in practice; see the following examples:

$$\begin{aligned} \boldsymbol{a} \cdot (\boldsymbol{a}_1 \wedge \boldsymbol{a}_2) &= \boldsymbol{a} \cdot \boldsymbol{a}_1 \boldsymbol{a}_2 - \boldsymbol{a} \cdot \boldsymbol{a}_2 \boldsymbol{a}_1, \\ \boldsymbol{a} \cdot (\boldsymbol{a}_1 \wedge \boldsymbol{a}_2 \wedge \boldsymbol{a}_3) &= \boldsymbol{a} \cdot \boldsymbol{a}_1 \boldsymbol{a}_2 \wedge \boldsymbol{a}_3 - \boldsymbol{a} \cdot \boldsymbol{a}_2 \boldsymbol{a}_1 \wedge \boldsymbol{a}_3 + \boldsymbol{a} \cdot \boldsymbol{a}_3 \boldsymbol{a}_1 \wedge \boldsymbol{a}_2. \end{aligned} \tag{2.49}$$

The generalization of Eq. (2.48) will be discussed in detail in later section.

2.3.2 Projections and Rejections

Let be $\boldsymbol{A}_r$ a non-degenerated r-blade or hyperplane in G_n, then any vector $\boldsymbol{v} \in \mathbb{R}^n$ has a *projection* (parallel component) onto the space of $\boldsymbol{A}_r$ defined by

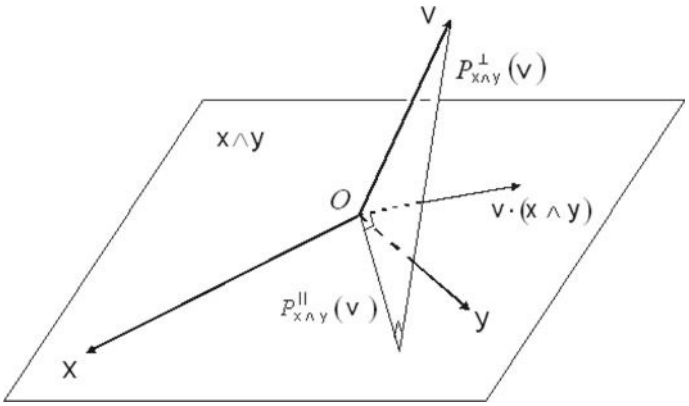


Fig. 2.3 Projection, rejection, and inner product with respect to $\boldsymbol{A}_2 = \boldsymbol{x} \wedge \boldsymbol{y}$

$$P^{\parallel}_{\boldsymbol{A}_r} = (\boldsymbol{v} \cdot \boldsymbol{A}_r)\boldsymbol{A}_r^{-1}, \tag{2.50}$$

and a *rejection* (perpendicular component) from the space of $\boldsymbol{A}_r$ defined by

$$P^{\perp}_{\boldsymbol{A}_r} = (\boldsymbol{v} \wedge \boldsymbol{A}_r)\boldsymbol{A}_r^{-1}. \tag{2.51}$$

Therefore, $\boldsymbol{v}$ can be *splitted* in terms of a projection and a *rejection* as follows:

$$\boldsymbol{v} = \boldsymbol{v}_{\parallel} + \boldsymbol{v}_{\perp} = P^{\parallel}_{\boldsymbol{A}_r} + P^{\perp}_{\boldsymbol{A}_r}, \tag{2.52}$$

which is an orthogonal decomposition of $\boldsymbol{v}$ with respect to $\boldsymbol{A}_r$. Figure 2.3 shows the projection, rejection, and inner product of the vector $\boldsymbol{v}$ with respect to the space (plane) $\boldsymbol{A}_2 = \boldsymbol{x} \wedge \boldsymbol{y}$. Note that the vector $\boldsymbol{v} \cdot \boldsymbol{x} \wedge \boldsymbol{y}$ is perpendicular to the vector $P^{\parallel}_{\boldsymbol{A}_2}$.

2.3.3 Projective Split

The idea of the *projective split* was introduced by Hestenes [12] in order to connect *projective geometry* and *metric geometry*. This is done by associating the even subalgebra of G_{n+1} with the geometric algebra of the next lower dimension, G_n. The multivector basis of G_{n+1} and G_n is given by $\gamma_1, \gamma_2, \ldots, \gamma_{n+1}$ and $e_1, e_2, \ldots, e_n$, respectively. One can define a mapping between the spaces by choosing a preferred direction in G_{n+1}, γ_{n+1}. Then, by taking the geometric product of a vector $\mathbf{X} \in G_{n+1}$ and γ_{n+1},

$$\mathbf{X}\gamma_{n+1} = \mathbf{X}\cdot\gamma_{n+1} + \mathbf{X} \wedge \gamma_{n+1} = \mathbf{X}\cdot\gamma_{n+1}\left(1 + \frac{\mathbf{X} \wedge \gamma_{n+1}}{\mathbf{X}\cdot\gamma_{n+1}}\right), \tag{2.53}$$

the vector $\boldsymbol{x} \in G_n$ can be associated with the bivector $\frac{\mathbf{X} \wedge \gamma_{n+1}}{\mathbf{X} \gamma_{n+1}} \in G_{n+1}$. This result can be projectively interpreted as the pencil of all lines passing though the point γ_{n+1}. In physics, the projective split is called the *space–time split*, and it relates a space–time system G_4 with a Minkowski metric to an observable system G_3 with a Euclidean metric.

2.3.4 Generalized Inner Product

Multivector computing involving inner products is easier if one employs the next equality for the generalized inner product of two blades $\boldsymbol{A}_r = \boldsymbol{a}_1 \wedge \boldsymbol{a}_2 \wedge \cdots \wedge \boldsymbol{a}_r$, and $\boldsymbol{B}_s = \boldsymbol{b}_1 \wedge \boldsymbol{b}_2 \wedge \cdots \wedge \boldsymbol{b}_s$:

$$\boldsymbol{A}_r \cdot \boldsymbol{B}_s = \begin{cases} ((\boldsymbol{a}_1 \wedge \boldsymbol{a}_2 \wedge \cdots \wedge \boldsymbol{a}_r) \cdot \boldsymbol{b}_1) \cdot (\boldsymbol{b}_2 \wedge \boldsymbol{b}_3 \wedge \cdots \wedge \boldsymbol{b}_s) & \text{if } r \geq s \\ (\boldsymbol{a}_1 \wedge \boldsymbol{a}_2 \wedge \cdots \wedge \boldsymbol{a}_{r-1}) \cdot (\mathbf{a}_r \cdot (\boldsymbol{b}_1 \wedge \boldsymbol{b}_2 \wedge \cdots \wedge \boldsymbol{b}_s)) & \text{if } r < s. \end{cases} \tag{2.54}$$

For *right contraction* use the following equation:

$$(\boldsymbol{a}_1 \wedge \boldsymbol{a}_2 \wedge \cdots \wedge \boldsymbol{a}_r) \cdot \boldsymbol{b}_1 = \sum_{i=1}^{r} (-1)^{r-i} \mathbf{a}_1 \wedge \cdots \wedge \mathbf{a}_{i-1} \wedge (\boldsymbol{a}_i \cdot \boldsymbol{b}_1) \wedge \boldsymbol{a}_{i+1} \wedge \cdots \wedge \boldsymbol{a}_r \tag{2.55}$$

and for *left contraction*

$$\mathbf{a}_r \cdot (\boldsymbol{b}_1 \wedge \boldsymbol{b}_2 \wedge \cdots \wedge \boldsymbol{b}_s) = \sum_{i=1}^{s} (-1)^{i-1} \boldsymbol{b}_1 \wedge \cdots \wedge \boldsymbol{b}_{i-1} \wedge (\boldsymbol{a}_r \cdot \boldsymbol{b}_i) \wedge \boldsymbol{b}_{i+1} \wedge \cdots \wedge \boldsymbol{b}_s. \tag{2.56}$$

From these equations, we can see that the inner product is not commutative for general multivectors. Indeed, if, in particular, $\mathbf{a}$, $\boldsymbol{b}$, and $\mathbf{c}$ are 1-vectors, then from (2.56),

$$\mathbf{a} \cdot (\boldsymbol{b} \wedge \mathbf{c}) = (\mathbf{a} \cdot \boldsymbol{b})\mathbf{c} - (\mathbf{a} \cdot \mathbf{c})\boldsymbol{b}, \tag{2.57}$$

and from Eq. (2.55),

$$(\boldsymbol{b} \wedge \mathbf{c}) \cdot \boldsymbol{a} = \boldsymbol{b}(\mathbf{c} \cdot \boldsymbol{a}) - (\boldsymbol{b} \cdot \boldsymbol{a})\mathbf{c} = -(\mathbf{a} \cdot \boldsymbol{b})\mathbf{c} + (\mathbf{a} \cdot \mathbf{c})\boldsymbol{b}. \tag{2.58}$$

Thus, the inner product of a vector $\boldsymbol{a}$ and the bivector $\boldsymbol{B} = \boldsymbol{b} \wedge \boldsymbol{c}$ fulfills

$$\boldsymbol{a} \cdot \boldsymbol{B} = -\boldsymbol{B} \cdot \boldsymbol{a}. \tag{2.59}$$

Let us now consider the inner product for the case $r < s$, given the bivector $\boldsymbol{A}_2 = \boldsymbol{a}_1 \wedge \boldsymbol{a}_2$ and the trivector $\boldsymbol{B}_3 = \boldsymbol{b}_1 \wedge \boldsymbol{b}_2 \wedge b_3$. Then, applying Eq. (2.56) from the left, we should get a vector $\boldsymbol{b}$:

$$
\begin{aligned}
\boldsymbol{A}_2 \cdot \boldsymbol{B}_3 &= \langle \boldsymbol{A}_2 \cdot \boldsymbol{B}_3 \rangle_{|2-3|} = \langle \boldsymbol{A}_2 \cdot \boldsymbol{B}_3 \rangle_1, \\
&= (\boldsymbol{a}_1 \wedge \boldsymbol{a}_2) \cdot (\boldsymbol{b}_1 \wedge \boldsymbol{b}_2 \wedge \boldsymbol{b}_3) = \boldsymbol{a}_1 \cdot [\boldsymbol{a}_2 \cdot (\boldsymbol{b}_1 \wedge \boldsymbol{b}_2 \wedge \boldsymbol{b}_3)], \\
&= \boldsymbol{a}_1 \cdot [(\boldsymbol{a}_2 \cdot \boldsymbol{b}_1)\boldsymbol{b}_2 \wedge \boldsymbol{b}_3 - (\boldsymbol{a}_2 \cdot \boldsymbol{b}_2)\boldsymbol{b}_1 \wedge \boldsymbol{b}_3 + (\boldsymbol{a}_2 \cdot \boldsymbol{b}_3)\boldsymbol{b}_1 \wedge \boldsymbol{b}_2], \\
&= (\boldsymbol{a}_2 \cdot \boldsymbol{b}_2)[(\boldsymbol{a}_1 \cdot \boldsymbol{b}_1)\boldsymbol{b}_3 - (\boldsymbol{a}_1 \cdot \boldsymbol{b}_3)\boldsymbol{b}_1] + \\
&\quad + (\boldsymbol{a}_2 \cdot \boldsymbol{b}_3)[(\boldsymbol{a}_1 \cdot \boldsymbol{b}_1)\boldsymbol{b}_2 - (\boldsymbol{a}_1 \cdot \boldsymbol{b}_2)\boldsymbol{b}_1], \\
&= [(\boldsymbol{a}_1 \cdot \boldsymbol{b}_3)[(\boldsymbol{a}_2 \cdot \boldsymbol{b}_2) - (\boldsymbol{a}_1 \cdot \boldsymbol{b}_2)(\boldsymbol{a}_2 \cdot \boldsymbol{b}_3)]\boldsymbol{b}_1 + \\
&\quad + [(\boldsymbol{a}_1 \cdot \boldsymbol{b}_1)[(\boldsymbol{a}_2 \cdot \boldsymbol{b}_3) - (\boldsymbol{a}_1 \cdot \boldsymbol{b}_3)(\boldsymbol{a}_2 \cdot \boldsymbol{b}_1)]\boldsymbol{b}_2 + \\
&\quad + [(\boldsymbol{a}_1 \cdot \boldsymbol{b}_2)[(\boldsymbol{a}_2 \cdot \boldsymbol{b}_1) - (\boldsymbol{a}_1 \cdot \boldsymbol{b}_1)(\boldsymbol{a}_2 \cdot \boldsymbol{b}_2)]\boldsymbol{b}_3, \\
&= \beta_1 \boldsymbol{b}_1 + \beta_2 \boldsymbol{b}_2 + \beta_3 \boldsymbol{b}_3 = \boldsymbol{b}.
\end{aligned} \tag{2.60}
$$

The projective geometric interpretation of this result tells us that the inner product of a line $\boldsymbol{A}_2$ with the plane $\boldsymbol{B}_3$ should be a vector $\boldsymbol{b}$ or point lying on the plane. Note that the point $\boldsymbol{b}$ is spanned as a linear combination of the vector basis that spans the plane $\boldsymbol{B}_3$. Later, we will see that in an algebra of incidence of the projective geometry, the meet operation (inner product with a dual multivector) computes the incidence point of a line through a plane.

Let us consider two other examples of the conformal geometric algebra $G_{4,1}$. Given the Euclidean pseudoscalar $I_E = e_1 \wedge e_2 \wedge e_3$ and the Minkowski plane $E = e_4 \wedge e_5$, let us compute the square of the pseudoscalar $I = I_E \wedge E = e_1 \wedge e_2 \wedge e_3 \wedge e_4 \wedge e_5$:

$$
\begin{aligned}
I^2 &= (I_E \wedge E) \cdot (I_E \wedge E) = -[(I_E \wedge (E \cdot I_E)] \cdot E + [(I_E \cdot I_e) \wedge E] \cdot E, \\
&= (I_E^2 \wedge E) \cdot E = ((-1) \cdot E) \cdot E = -E^2 = -1,
\end{aligned} \tag{2.61}
$$

since E and I_E are orthogonal, $E \cdot I_E = 0$.

Let us now consider Eq. (2.64) $G_{n+1,1}$ given in [2], which claims that the component of the sphere s in $\mathbb{R}^n$ is given by

$$
\boldsymbol{s} = P_E^{\perp}(s) = (s \wedge E)E = s + (s \cdot e_0)e. \tag{2.62}
$$

Since the Minkowski plane can also be expressed as $E = e_\infty \wedge e_0$, where $e_\infty = e_{n+1} + e_{n+2}$, $e_0 = 0.5(e_{n+1} - e_{n+2})$, and $E^2 = 1$, we proceed by computing the inner product from the right using Eq. (2.55) and the equality $\boldsymbol{B} \cdot (\boldsymbol{a} \wedge \boldsymbol{b}) = \boldsymbol{B} \cdot a \cdot b$, namely,

$$
\begin{aligned}
s &= s \cdot 1 = (sE)E = (s \cdot E)E + (s \wedge E)E = P_E^{\parallel}(s) + P_E^{\perp}(s), \\
&= P_E^{\perp}(s) = (s \wedge E)E = (s \wedge e_\infty \wedge e_0) \wedge (e_\infty \wedge e_0) + (s \wedge e_\infty \wedge e_0) \cdot (e_\infty \wedge e_0), \\
&= [(s \wedge e_\infty \wedge e_0) \cdot e_\infty] \cdot e_0, \\
&= [(s \wedge e_\infty) \wedge (e_0 \cdot e_\infty) - s \wedge (e_\infty \cdot e_\infty) \wedge e_0 + (s \cdot e_\infty) \wedge e_\infty \wedge e_0] \cdot e_0, \\
&= -(s \wedge \epsilon_\infty) \cdot e_0 = -[s \wedge (e_\infty \cdot e_0) - (s \cdot e_0) \wedge e_\infty], \\
&= -[-s - (s \cdot e_0) \wedge e_\infty] = s + (s \cdot e_0) \wedge e_\infty.
\end{aligned} \tag{2.63}
$$

Let us see how we can compute the meet $\cap$ (incidence algebra intersection) and the generalized inner product. Consider the meet of the k-vectors $\boldsymbol{C}_q$, $\boldsymbol{A}_s \in G_n$

$$\boldsymbol{C}_q \cap \boldsymbol{A}_s = \boldsymbol{C}^*_{n-q} \cdot \boldsymbol{A}_s = (\boldsymbol{C}_q \boldsymbol{I}_n^{-1}) \cdot \boldsymbol{A}_s = \boldsymbol{B}_r \cdot \boldsymbol{A}_s, \tag{2.64}$$

where the meet operation is expressed as the inner product of the dual of $\boldsymbol{C}_q$, i.e., $\boldsymbol{C}^*_{n-q} = \boldsymbol{C}_q \boldsymbol{I}_n^{-1} = \boldsymbol{B}_r$, and $\boldsymbol{A}_s$. Thereafter, this inner product can be then easily computed using the generalized inner product of Eq. (2.54).

2.3.5 Geometric Product of Multivectors

In general, any two paired multivectors can be multiplied using the geometric product. Consider two homogeneous multivectors $\boldsymbol{A}_r$ and $\boldsymbol{B}_s$ of grade r and s, respectively. The geometric product of $\boldsymbol{A}_r$ and $\boldsymbol{B}_s$ can be written as

$$\boldsymbol{A}_r \boldsymbol{B}_s = \langle \boldsymbol{AB} \rangle_{r+s} + \langle \boldsymbol{AB} \rangle_{r+s-2} + \cdots + \langle \boldsymbol{AB} \rangle_{|r-s|}, \tag{2.65}$$

where $\langle \boldsymbol{AB} \rangle_t$ denotes the t-grade part the multivector $\boldsymbol{A}_r \boldsymbol{B}_s$. Note that

$$\langle \boldsymbol{A}_r \boldsymbol{B}_s \rangle_{|r-s|} \tag{2.66}$$

is a generalized contraction or inner product, and

$$\boldsymbol{A}_r \wedge \boldsymbol{B}_s \equiv \langle \boldsymbol{A}_r \boldsymbol{B}_s \rangle_{r+s} \tag{2.67}$$

is the *outer* or *wedge product* of the highest grade. The *inner product* of lowest grade $\langle \boldsymbol{AB} \rangle_0$ corresponds to a full contraction, or standard inner product. Since the elements of $\boldsymbol{A}_r \boldsymbol{B}_s$ are of a different grade, $\boldsymbol{A}_r \boldsymbol{B}_s$ is thus an *inhomogeneous multivector*. As illustration, consider $\boldsymbol{ab} = \langle \boldsymbol{ab} \rangle_0 + \langle \boldsymbol{ab} \rangle_2 = \boldsymbol{a} \cdot \boldsymbol{b} + \boldsymbol{a} \wedge \boldsymbol{b}$ and the geometric product of $\boldsymbol{A} = 5e_3 + 3e_1 e_2$ and $\boldsymbol{b} = 9e_2 + 7e_3$:

$$\begin{aligned} \boldsymbol{Ab} &= \langle \boldsymbol{Ab} \rangle_0 + \langle \boldsymbol{Ab} \rangle_1 + \langle \boldsymbol{Ab} \rangle_2 + \langle \boldsymbol{Ab} \rangle_3, \\ &= 35(e_3)^2 + 27e_1(e_2)^2 + 45e_3 e_2 + 21e_1 e_2 e_3, \\ &= 35 + 27e_1 - 45e_2 e_3 + 21I. \end{aligned} \tag{2.68}$$

In the following sections, expressions of grade 0 will be written ignoring their subindex, that is, $\langle \boldsymbol{ab} \rangle_0 = \langle \boldsymbol{ab} \rangle$.

We can see from Eq. (2.66) that the inner product of a scalar α with a homogeneous multivector $\boldsymbol{A}_r$ has a special treatment: $\boldsymbol{A} \cdot \alpha = \alpha \cdot \boldsymbol{A} = 0$. However, it is not the case for the outer product: $\boldsymbol{A} \wedge \alpha = \alpha \wedge \boldsymbol{A} = \langle \alpha \boldsymbol{A} \rangle_r \equiv \alpha \boldsymbol{A}$. We make the observation that software like GABLE or CLICAL does not have this exceptional case for the inner

product. Now, the inner and outer products of any two general multivectors will be obtained applying the left and right distributive laws over their homogeneous parts and then using Eqs. (2.66) and (2.67). Note that the definition in (2.66) is not in contradiction to (2.32). Analogously, (2.67) and (2.33) are consistent. From (2.66), it can be said that the inner product $\boldsymbol{A}_r \cdot \boldsymbol{B}_s$ *lowers the grade of* $\boldsymbol{A}_r$ *by s units* when $r \geq s > 0$, and from Eq. (2.67) that the outer product $\boldsymbol{A}_r \wedge \boldsymbol{B}_s$ *raises the grade of* $\boldsymbol{A}_r$ *by s units* for every $r, s \geq 0$.

2.3.6 Contractions and the Derivation

Let us consider a vector and a bivector $\boldsymbol{v}, \boldsymbol{A} \in G_3$. The vector $\boldsymbol{v}$ is tilted by an angle θ out of the plane of a bivector $\boldsymbol{A}$; see Fig. 2.4. Let $\boldsymbol{x}$ be the orthogonal projection of $\boldsymbol{v}$ in the plane, then $|\boldsymbol{x}| = |\boldsymbol{v}| \cos\theta$.

In this setting, Lounesto [11] introduced the following notion: The right *contraction* of the bivector $\boldsymbol{A}$ by the vector $\boldsymbol{v}$ is a vector $\boldsymbol{y} = \boldsymbol{A} \vdash \boldsymbol{v}$ that is orthogonal to $\boldsymbol{x}$ and lies on the plane of the bivector $\boldsymbol{A}$ such that

$$\begin{aligned} &|\boldsymbol{y}| = |\boldsymbol{A}||\boldsymbol{x}|, \\ &\boldsymbol{y} \perp \boldsymbol{x} \quad \text{and} \quad \boldsymbol{x} \wedge \boldsymbol{y} \parallel \boldsymbol{A}. \end{aligned} \tag{2.69}$$

According to Eq. (2.59),

$$\boldsymbol{v} \dashv \boldsymbol{A} = -\boldsymbol{A} \vdash \boldsymbol{v}, \tag{2.70}$$

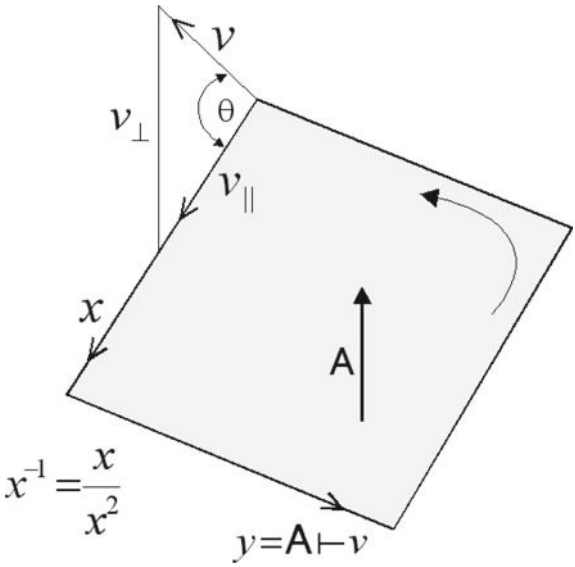


Fig. 2.4 Projection, rejection, and inner product with respect to $\boldsymbol{A}_2 = \boldsymbol{a}_1 \wedge \boldsymbol{a}_2$

that is, the left and right contractions have opposite signs. In Fig. 2.4, we see that the area of the rectangle can be computed in terms of the inverse of $\boldsymbol{x}$, namely, $|\boldsymbol{A}| = |\boldsymbol{x}^{-1}||\boldsymbol{y}|$, where $\boldsymbol{x}^{-1} = \frac{\boldsymbol{x}}{\boldsymbol{x}^2}$. If we write $\boldsymbol{v}_{\parallel} = \boldsymbol{x}$ and $\boldsymbol{v}_{\perp} = \boldsymbol{x} - \boldsymbol{v}_{\parallel}$, the vector $\boldsymbol{v}$ can be expressed in terms of its parallel and perpendicular components:

$$\boldsymbol{v} = P_{\boldsymbol{A}}^{\parallel} + P_{\boldsymbol{A}}^{\perp} = \boldsymbol{v}_{\parallel} + \boldsymbol{v}_{\perp} = (\boldsymbol{v} \dashv \boldsymbol{A})\boldsymbol{A}^{-1} + (\boldsymbol{v} \wedge \boldsymbol{A})\boldsymbol{A}^{-1}, \tag{2.71}$$

where $\boldsymbol{A}^{-1} = \boldsymbol{A}/\boldsymbol{A}^2, \boldsymbol{A}^2 = -|\boldsymbol{A}|^2$. According to Eqs. (2.44) and (2.45), the geometric product of $\boldsymbol{v}$ and the bivector $\boldsymbol{A}$ can also be rewritten as

$$\boldsymbol{v}\boldsymbol{A} = \boldsymbol{v} \dashv \boldsymbol{A} + \boldsymbol{v} \wedge \boldsymbol{A} = \frac{1}{2}(\boldsymbol{v}\boldsymbol{A} - \boldsymbol{A}\boldsymbol{v}) + \frac{1}{2}(\boldsymbol{v}\boldsymbol{A} + \boldsymbol{A}\boldsymbol{v}). \tag{2.72}$$

The left contraction can also be computed using the notion of duality, given the vectors and pseudoscalar $\boldsymbol{a}$, $\boldsymbol{b}$, $I \in G_3$:

$$\boldsymbol{a} \dashv \boldsymbol{b} = [\boldsymbol{a} \wedge (\boldsymbol{b}I)]I^{-1}. \tag{2.73}$$

This equation means that the left contraction is dual to the wedge product. Later, we will discuss at length the concept of duality and the related operations of meet and join of the incidence algebra.

Now consider in our next formula the vectors $\boldsymbol{v}, \boldsymbol{b}$, and the k-vector $\boldsymbol{A}_k \in G_3$:

$$\boldsymbol{v} \dashv (\boldsymbol{A}_k\boldsymbol{b}) = (\boldsymbol{v} \dashv \boldsymbol{A}_k)\boldsymbol{b} + (-1)^k\boldsymbol{A}_k(\boldsymbol{v} \dashv \boldsymbol{b}). \tag{2.74}$$

We see that the left contraction by a vector is a *derivation* in the geometric algebra.

In general, the product of a vector $\boldsymbol{v}$ and an arbitrary r-vector $\boldsymbol{A}_r$ can be decomposed into a sum of the left contraction and the wedge product. Thus using Eqs. (2.44) and (2.45),

$$\begin{aligned} \boldsymbol{v}\boldsymbol{A}_r &= \boldsymbol{v} \dashv \boldsymbol{A}_r + \boldsymbol{v} \wedge \boldsymbol{A}_r, \\ &= \frac{1}{2}\left(\boldsymbol{v}\boldsymbol{A}_r - (-1)^r\boldsymbol{A}_r\boldsymbol{v}\right) + \frac{1}{2}(\boldsymbol{v}\boldsymbol{A}_r + (-1)^r\boldsymbol{A}_r\boldsymbol{v}\Big). \end{aligned} \tag{2.75}$$

As we have seen in this section, the right $\dashv$ and left $\vdash$ contractions can be computed simply using Eqs. (2.55) and (2.56) of the generalized inner product (2.54). Therefore, in order to avoid using an excessive number of operator symbols, in this book we will not use the operators $\dashv$ and $\vdash$ when carrying out contractions.

2.3.7 Hodge Dual

The Hodge star operator establishes a correspondence between the space $\bigwedge^{k} V^n$ of k-vectors with the space $\bigwedge^{n-k} V^n$ of $(n-k)$-vectors. The dimension of the former space is $\binom{n}{k}$ and that of the latter space is $\binom{n}{n-k}$. Due to the symmetry of the binomial coefficients, the dimensions of both spaces are equal. The mapping between spaces of the same dimension is an isomorphism, which, in the case of the Hodge duality, exploits the inner product and orientation of the vector space. The image of a k-vector under this isomorphism is known as its *Hodge dual* of the k-vector.

The Hodge star operator on an oriented inner product space V^n is a linear operator on the exterior algebra of V^n. This operator interchanges the subspaces of k-vectors and $(n-k)$-vectors, as we show next. Given an oriented orthonormal basis $e_1, e_2, \ldots, e_n$, the Hodge dual of a k-vector is computed as follows:

$$\star\,(e_1 \wedge e_2 \wedge \cdots \wedge e_k) = e_{k+1} \wedge e_{k+2} \wedge \cdots \wedge e_n. \tag{2.76}$$

As an illustration, let us apply the Hodge dual to the vector $\boldsymbol{y} \in \mathbb{R}^3$:

$$\begin{aligned} \boldsymbol{Y} = \star\boldsymbol{y} &= \star(y_1 e_1 + y_2 e_2 + y_3 e_3), \\ &= y_1 e_2 \wedge e_3 + y_2 e_3 \wedge e_1 + y_3 e_1 \wedge e_2 \in \bigwedge^{2} R^3 \end{aligned} \tag{2.77}$$

and now compute the wedge of a vector $\boldsymbol{x} \in \mathbb{R}^3$ with the bivector $\boldsymbol{Y} = \star\boldsymbol{y} \in \bigwedge^{2} R^3$:

$$\boldsymbol{x} \wedge \boldsymbol{Y} = \boldsymbol{x} \wedge \star\boldsymbol{y} = (\boldsymbol{x} \cdot \boldsymbol{y}) e_1 \wedge e_2 \wedge e_3, \tag{2.78}$$

or the wedge product of a bivector $\boldsymbol{X}$ with the Hodge dual of a bivector $\boldsymbol{Y}$:

$$\boldsymbol{X} \wedge \star\boldsymbol{Y} = \langle \boldsymbol{X}, \boldsymbol{Y} \rangle e_1 \wedge e_2 \wedge e_3. \tag{2.79}$$

Using the Hodge duality, we can establish in $\mathbb{R}^3$ the relationship between the cross product and the wedge product as follows:

$$\begin{aligned} \boldsymbol{x} \wedge \boldsymbol{y} &= \star(\boldsymbol{x} \times \boldsymbol{y}), \\ \boldsymbol{x} \times \boldsymbol{y} &= \star(\boldsymbol{x} \wedge \boldsymbol{y}). \end{aligned} \tag{2.80}$$

In the geometric algebra G_3, the Hodge dual is computed using the reversion of the multivector

$$\star\boldsymbol{A} = \widetilde{\boldsymbol{A}} I_3, \tag{2.81}$$

where $I_3 = e_1 \wedge e_2 \wedge e_3$ is the pseudoscalar. The reversion is defined in Eq. (2.92). For example, the Hodge dual of a vector $\boldsymbol{x}$ and a bivector $\boldsymbol{X}$ are computed as

$$\begin{aligned} \star \boldsymbol{x} &= \widetilde{\boldsymbol{x}} I_3 = \boldsymbol{x} I_3 = \boldsymbol{X}, \\ \star \boldsymbol{X} &= \widetilde{\boldsymbol{X}} I_3 = -\boldsymbol{X} I_3 = \boldsymbol{x}. \end{aligned} \tag{2.82}$$

Finally, the relationship between the cross product and the wedge product is written as

$$\begin{aligned} \boldsymbol{x} \wedge \boldsymbol{y} &= \star(\boldsymbol{x} \times \boldsymbol{y}) = (\boldsymbol{x} \times \boldsymbol{y}) I_3, \\ \boldsymbol{x} \times \boldsymbol{y} &= -(\boldsymbol{x} \wedge \boldsymbol{y}) I_3. \end{aligned} \tag{2.83}$$

2.3.8 Dual Blades and Duality in the Geometric Product

A fundamental concept algebraically related to the unit pseudoscalar I is that of *duality*. In a geometric algebra G_n, we find dual multivectors and dual operations. The dual of a multivector $\boldsymbol{A} \in G_n$ is defined as follows:

$$\boldsymbol{A}^* = \boldsymbol{A} I_n^{-1}, \tag{2.84}$$

where I_n^{-1} differs from I_n at most by a sign. Note that, in general, I^{-1} might not necessarily commute with $\boldsymbol{A}$.

The multivector bases of a geometric algebra G_n have 2^n basis elements. It can be shown that the second half is the dual of the first half. For example, in G_3 the dual of the scalar is the pseudoscalar, and the dual of a vector is a bivector $e_{23} = I e_1$. In general, the dual of an r-blade is an $(n - r)$-blade.

The operations related directly to the Clifford product are the inner and outer products, which are dual to one another. This can be written as follows:

$$(\boldsymbol{x} \cdot \boldsymbol{A}) I_n = \boldsymbol{x} \wedge (\boldsymbol{A} I_n), \tag{2.85}$$

$$(\boldsymbol{x} \wedge \boldsymbol{A}) I_n = \boldsymbol{x} \cdot (\boldsymbol{A} I_n), \tag{2.86}$$

where $\boldsymbol{x}$ is any vector and $\boldsymbol{A}$ any multivector.

By using the ideas of duality, we are then able to relate the inner product to incidence operators in the following manner. In an n-dimensional space, suppose we have an r-vector $\boldsymbol{A}$ and an s-vector $\boldsymbol{B}$ where the dual of $\boldsymbol{B}$ is given by $\boldsymbol{B}^* = \boldsymbol{B} I^{-1} \equiv \boldsymbol{B} \cdot I^{-1}$. Since $\boldsymbol{B} I^{-1} = \boldsymbol{B} \cdot I^{-1} + \boldsymbol{B} \wedge I^{-1}$, we can replace the geometric product by the inner product alone (in this case, the outer product equals zero, and there can be no $(n + 1)$-D vector). Now, using the identity

$$\boldsymbol{A}_r \cdot (\boldsymbol{B}_s \cdot \boldsymbol{C}_t) = (\boldsymbol{A}_r \wedge \boldsymbol{B}_s) \cdot \boldsymbol{C}_t \qquad \text{for} \qquad r + s \leq t, \tag{2.87}$$

we can write

$$\boldsymbol{A}\cdot(\boldsymbol{B}I^{-1}) = \boldsymbol{A}\cdot(\boldsymbol{B}\cdot I^{-1}) = (\boldsymbol{A}\wedge\boldsymbol{B})\cdot I^{-1} = (\boldsymbol{A}\wedge\boldsymbol{B})I^{-1}. \tag{2.88}$$

This expression can be rewritten using the definition of the dual as follows:

$$\boldsymbol{A}\cdot\boldsymbol{B}^{*} = (\boldsymbol{A}\wedge\boldsymbol{B})^{*}. \tag{2.89}$$

The equation shows the relationship between the inner and outer products in terms of the duality operator. Now, if $r+s=n$, then $\boldsymbol{A}\wedge\boldsymbol{B}$ is of grade n and is therefore a pseudoscalar. Using Eq. (2.88), we can employ the involved pseudoscalar in order to get an expression in terms of a bracket:

$$\begin{aligned}\boldsymbol{A}\cdot\boldsymbol{B}^{*} &= (\boldsymbol{A}\wedge\boldsymbol{B})^{*} = (\boldsymbol{A}\wedge\boldsymbol{B})I^{-1} = ([\boldsymbol{A}\wedge\boldsymbol{B}]I)I^{-1},\\ &= [\boldsymbol{A}\wedge\boldsymbol{B}].\end{aligned} \tag{2.90}$$

We see, therefore, that the bracket relates the inner and outer products to nonmetric quantities.

2.4 Multivector Operations

In this section, we will define a number of right operations that act on multivectors and return multivectors.

2.4.1 Involution, Reversion, and Conjugation Operations

For an r-grade multivector $\boldsymbol{A}_r = \sum_{i=0}^{r}\langle\boldsymbol{A}_r\rangle_i$, the following operations are defined:

$$\text{Grade involution: } \widehat{\boldsymbol{A}}_r = \sum_{i=0}^{r}(-1)^i\langle\boldsymbol{A}_r\rangle_i, \tag{2.91}$$

$$\text{Reversion: } \widetilde{\boldsymbol{A}}_r = \sum_{i=0}^{r}(-1)^{\frac{i(i-1)}{2}}\langle\boldsymbol{A}_r\rangle_i, \tag{2.92}$$

$$\begin{aligned}\text{Conjugate: } \langle\boldsymbol{A}_r\rangle_k^{\dagger} &= (\boldsymbol{a}_1\wedge\boldsymbol{a}_2\wedge\cdots\wedge\boldsymbol{a}_k)^{\dagger},\\ &= \boldsymbol{a}_k^{\dagger}\wedge\boldsymbol{a}_{k-1}^{\dagger}\wedge\cdots\wedge\boldsymbol{a}_1^{\dagger},\end{aligned} \tag{2.93}$$

$$\text{Clifford conjugation: } \overline{\boldsymbol{A}}_r = \widetilde{\widehat{\boldsymbol{A}}}_r = \sum_{i=0}^{r}(-1)^{\frac{i(i+1)}{2}}\langle\boldsymbol{A}_r\rangle_i. \tag{2.94}$$

The grade involution simply negates the odd-grade blades of a multivector. The reversion can also be obtained by reversing the order of basis vectors making up the blades in a multivector and then rearranging them in their original order using the anticommutativity of the Clifford product.

Let us explain the case of conjugation; let $\boldsymbol{A} \in \{1, \ldots, p+q\}$ be the set of vectors $e_{\boldsymbol{A}} \in G_{p,q}$; the conjugate of $e_{\boldsymbol{A}}$, denoted by $e_{\boldsymbol{A}}^{\dagger}$, is defined as

$$e_{\boldsymbol{A}}^{\dagger} = (-1)^r \tilde{e}_{\boldsymbol{A}}, \quad r := \texttt{grade of } e_{\boldsymbol{A}}. \tag{2.95}$$

Thus, while $\tilde{\boldsymbol{a}}_i = \boldsymbol{a}_i$, $\boldsymbol{a}_i^{\dagger}$ is not necessarily equal to $\boldsymbol{a}_i$, this is because for $e_i \in G_{p,q}$, $e_i^{\dagger} = -e_i$ if $p < i < p+q$.

The computation of the *magnitude* or modulus of a blade can be done using the reversion operation, as follows:

$$||\boldsymbol{A}|| = \langle \widetilde{\boldsymbol{A}} \boldsymbol{A} \rangle_0^{\frac{1}{2}} = ||\langle \boldsymbol{A} \rangle_k||^2. \tag{2.96}$$

Accordingly, the magnitude of a multivector $\boldsymbol{M}$ reads

$$\begin{aligned} ||\boldsymbol{M}|| &= \langle \widetilde{\boldsymbol{M}} \boldsymbol{M} \rangle_0^{\frac{1}{2}}, \\ &= (||\langle \boldsymbol{M} \rangle_0||^2 + ||\langle \boldsymbol{M} \rangle_1||^2 \\ &\qquad + ||\langle \boldsymbol{M} \rangle_2||^2 + \cdots + ||\langle \boldsymbol{M} \rangle_n||^2)^{\frac{1}{2}}, \\ &= \sqrt{\sum_{r=0}^{n} ||\langle \boldsymbol{M} \rangle_r||^2}. \end{aligned} \tag{2.97}$$

In particular, for an *r-vector* $\boldsymbol{A}_r$ of the form $\boldsymbol{A}_r = \mathbf{a}_1 \wedge \boldsymbol{a}_2 \wedge \cdots \wedge \boldsymbol{a}_r$: $\boldsymbol{A}_r^{\dagger} = (\mathbf{a}_1 \cdots \mathbf{a}_{r-1} \mathbf{a}_r)^{\dagger} = \mathbf{a}_r \mathbf{a}_{r-1} \cdots \boldsymbol{a}_1$ and thus $\boldsymbol{A}_r^{\dagger} \boldsymbol{A}_r = \mathbf{a}_1^2 \, \mathbf{a}_2^2 \cdots \boldsymbol{a}_r^2$, so, we will say that such an r-vector is null if and only if it has a null vector for a factor. If in such a *factorization* of $\boldsymbol{A}_r$ p, q, and s factors square in a positive number, negative, and zero, respectively, we will say that $\boldsymbol{A}_r$ is an r-vector with signature (p, q, s). In particular, if $s = 0$ such a *nonsingular* r-vector has a multiplicative inverse that can also be written using the reversion

$$\boldsymbol{A}^{-1} = (-1)^q \frac{\boldsymbol{A}^{\dagger}}{\boldsymbol{A}\boldsymbol{A}^{\dagger}} = (-1)^q \frac{\boldsymbol{A}^{\dagger}}{|\boldsymbol{A}|^2} = \frac{\boldsymbol{A}}{\boldsymbol{A}^2}. \tag{2.98}$$

In general, the *inverse* $\boldsymbol{A}^{-1}$ of a multivector $\boldsymbol{A}$, if it exists, is defined by the equation $\boldsymbol{A}^{-1}\boldsymbol{A} = 1$.

2.4.2 Join and Meet Operations

When we work with lines, planes, and spheres, however, it will clearly be necessary to employ operations to compute the *meets* (intersections) or *joins* (expansions) of geometric objects. For this, we will need a geometric means of performing the set-theory operations of intersection, $\cap$, and union, $\cup$. Herewith, $\cup$ and $\cap$ will stand for the algebra of incidence operations of join and meet, respectively.

If in an n-dimensional geometric algebra the r-vector A and the s-vector B do not have a common subspace (null intersection), one can define the *join* of both vectors as

$$\boldsymbol{C} = \boldsymbol{A} \cup \boldsymbol{B} = \boldsymbol{A} \wedge \boldsymbol{B}, \tag{2.99}$$

so that the join is simply the outer product (an $r + s$ vector) of the two vectors. However, if $\boldsymbol{A}$ and $\boldsymbol{B}$ have common blades, the join would not simply be given by the wedge but by the subspace the two vectors span. The operation join $\cup$ can be interpreted as a *common dividend of lowest grade* and is defined up to a scale factor. The join gives the pseudoscalar if $(r + s) \geq n$. We will use $\cup$ to represent the join only when the blades A and B have a common subspace; otherwise, we will use the ordinary exterior product, $\wedge$, to represent the join.

If there exists a k-vector $\boldsymbol{D}$ such that for $\boldsymbol{A}$ and $\boldsymbol{B}$ we can write $\boldsymbol{A} = \boldsymbol{A}'\boldsymbol{D}$ and $\boldsymbol{B} = \boldsymbol{B}'\boldsymbol{D}$ for some $\boldsymbol{A}'$ and $\boldsymbol{B}'$, then we can define the *intersection* or *meet* using the duality principle as follows:

$$\boldsymbol{D} \cdot \boldsymbol{I}^{-1} = \boldsymbol{D}^* = (\boldsymbol{A} \cap \boldsymbol{B})^* = \boldsymbol{A}^* \cup \boldsymbol{B}^*. \tag{2.100}$$

This is a beautiful result, telling us that the dual of the meet is given by the join of the duals. Since the dual of $\boldsymbol{A} \cap \boldsymbol{B}$ will be taken with respect to the *join* of $\boldsymbol{A}$ and $\boldsymbol{B}$, we must be careful to specify which space we will use for the dual in Eq. (2.100). However, in most cases of practical interest, this join will indeed cover the entire space, and therefore we will be able to obtain a more useful expression for the meet using Eq. (2.89). Thus,

$$\boldsymbol{A} \cap \boldsymbol{B} = ((\boldsymbol{A} \cap \boldsymbol{B})^*)^* = (\boldsymbol{A}^* \cup \boldsymbol{B}^*)\boldsymbol{I} = (\boldsymbol{A}^* \wedge \boldsymbol{B}^*)(\boldsymbol{I}^{-1}\boldsymbol{I})\boldsymbol{I} = (\boldsymbol{A}^* \cdot \boldsymbol{B}). \tag{2.101}$$

The above concepts are discussed further in [8]. In the theory of Grassmann–Cayley algebra, the meet operation is computed using the so-called shuffle formula. This formula will be explained in Sect. 20.2.3 of Appendix in Chap. 20 [9].

2.4.3 Multivector-Valued Functions and the Inner Product

A multivector-valued function $f : \mathbb{R}^{p,q} \to G_{p,q}$, where $n = p + q$, has 2^n blade components

$$f(\boldsymbol{x}) = \sum_M f(\boldsymbol{x})_M e_M = \sum_{i=1}^{n} \langle f(\boldsymbol{x})\rangle_n, \tag{2.102}$$

where $M \in \{0, 1, 2, \ldots, 12, 23, \ldots, 123, 124, \ldots, 123, \ldots, (n-1)n\}$ contains all possible 2^n blade sub-indexes and $f(\boldsymbol{x})_M \in \mathbb{R}$ corresponds to the scalar accompanying each multivector base. Let us consider the complex conjugation in G_3

$$\begin{aligned}\tilde{f}(\boldsymbol{x}) = \sum_M f(\boldsymbol{x})_M \tilde{e}_M &= f(\boldsymbol{x})_0 e_0 + f(\boldsymbol{x})_1 e_1 + f(\boldsymbol{x})_2 e_2 + f(\boldsymbol{x})_3 e_3 - \\ &- f(\boldsymbol{x})_{23} e_{23} - f(\boldsymbol{x})_{32} e_{32} - f(\boldsymbol{x})_{12} e_{12} - f(\boldsymbol{x})_{123} e_{123}, \\ &= \langle f(\boldsymbol{x})\rangle_0 + \langle f(\boldsymbol{x})\rangle_1 - \langle f(\boldsymbol{x})\rangle_2 - \langle f(\boldsymbol{x})\rangle_3. \end{aligned} \tag{2.103}$$

Next, we define the inner product of $\mathbb{R}^n \to G_n$ functions f, g by

$$(f, g) = \sum_{\mathbb{R}^n} f(\boldsymbol{x})\widetilde{g(\boldsymbol{x})} d^n\boldsymbol{x}. \tag{2.104}$$

Let us consider the inner product of two functions in G_3:

$$(f, g)_{L^2(\mathbb{R}^3;G_3)} = \int_{\mathbb{R}^3} f(\boldsymbol{x})\widetilde{g(\boldsymbol{x})} d^3\boldsymbol{x} = \sum_{M,N} e_M \tilde{e}_N \int_{\mathbb{R}^3} f_M(\boldsymbol{x}) g_N(\boldsymbol{x}) d^3\boldsymbol{x}, \tag{2.105}$$

where $d^3\boldsymbol{x} = \frac{dx_1 \wedge dx_2 \wedge dx_3}{I_3}$. The scalar part of this equation corresponds to the L^2-norm

$$||f||^2_{L^2(\mathbb{R}^3;G_3)} = \langle (f, f)_{L^2(\mathbb{R}^3;G_3)}\rangle_0 = \sum_{\mathbb{R}^3} f(\boldsymbol{x}) * \tilde{f}(\boldsymbol{x}) d^3\boldsymbol{x} = \int_{\mathbb{R}^3} \sum_M f_M^2(\boldsymbol{x}) d^3\boldsymbol{x}.$$

The $L^2(\mathbb{R}^n; G_{n,0})$-norm is given by

$$||f||^2 = \langle (f, f)\rangle_0 = \sum_{\mathbb{R}^n} |f(\boldsymbol{x})|^2 d^n\boldsymbol{x}. \tag{2.106}$$

For the case of square-integrable functions on the sphere $L_2(S^{n-1})$, the inner product and the norm are

$$(f\,,g)_{L_2} = \int_{S^{n-1}} f(\boldsymbol{x})\widetilde{g(\boldsymbol{x})}dS(\boldsymbol{x}), \tag{2.107}$$

$$||f||^2 = 2^n \int_{S^{n-1}} \langle f(\boldsymbol{x})\widetilde{f(\boldsymbol{x})}\rangle_0 dS(\boldsymbol{x}), \tag{2.108}$$

where $dS(\boldsymbol{x})$ is the normalized Spin(n)-invariant measure on S^{n-1}.

2.4.4 The Multivector Integral

Let us consider $F(\boldsymbol{x})$ a multivalued function (field) of a vector variable $\boldsymbol{x}$ defined in a certain region of the Euclidean space E^n. If the function is only scalar or vector valued, it will be called a scalar of a vector fields, respectively. The Riemann integral of a multivector-valued function $F(\boldsymbol{x})$ is defined as follows:

$$\int_{E^n} F(\boldsymbol{x})|d\boldsymbol{x}| = \lim_{\substack{|\triangle x_j|\to 0\\ n\to\infty}} \sum_{j=1}^{n} F(x_j e_j)\triangle x_j, \tag{2.109}$$

where the quantity in brackets $|d\boldsymbol{x}|$ is used to make the integral grade preserving, because $d\boldsymbol{x}$ is a vector in a geometric algebra G_n. Thus, the integral can be discretized using the sum of quadrature expressions.

2.4.5 Convolution and Correlation of Scalar Fields

The filtering of a continuous signal $f : E^n \to \mathcal{C}$ is computed via the convolution of the signal with a filter $h : E^n \to \mathcal{C}$ as follows:

$$(h * f)(\boldsymbol{x}) = \int_{E^n} h(\boldsymbol{x}')f(\boldsymbol{x}-\boldsymbol{x}')d\boldsymbol{x}'. \tag{2.110}$$

The spatial correlation is defined by

$$(h \star f)(\boldsymbol{x}) = \int_{E^n} h(\boldsymbol{x}')f(\boldsymbol{x}+\boldsymbol{x}')d\boldsymbol{x}'. \tag{2.111}$$

Note that the convolution can be seen as a correlation with a filter that has been reflected with respect to its center.

2.4.6 Clifford Convolution and Correlation

Given a multivector field $\boldsymbol{F}$ and a multivector-valued filter $\boldsymbol{H}$, their Clifford convolution is defined in terms of the Clifford product of multivectors:

$$\begin{aligned}(\boldsymbol{H} *_l \boldsymbol{F})(\boldsymbol{x}) &= \int_{E^n} \boldsymbol{H}(\boldsymbol{x}')\boldsymbol{F}(\boldsymbol{x}-\boldsymbol{x}')|d\ x'|,\\ (\boldsymbol{F} *_r \boldsymbol{H})(\boldsymbol{x}) &= \int_{E^n} \boldsymbol{F}(\boldsymbol{x}-\boldsymbol{x}')\boldsymbol{H}(\boldsymbol{x}')|d\ x'|.\end{aligned} \tag{2.112}$$

Since the Clifford product is not commutative, we distinguish the application of the filter $\boldsymbol{H}$ from the left and from the right by using the subindex l and r, respectively. For the case of discrete fields, the convolution has to be discretized. If the application of the filter is done from the left, the discretized, left-convolution is given by

$$(\boldsymbol{H} *_l \boldsymbol{F})_{i,j,k} = \sum_{i=-d}^{d}\sum_{j=-d}^{d}\sum_{k=-d}^{d} \boldsymbol{H}_{r,s,t}\boldsymbol{F}_{i-r,j-s,k-t}, \tag{2.113}$$

where a 3D uniform grid is used where $i, j, k, r, s, t \in \mathbb{Z}$, (i, j, k) denotes the grid nodes and d^3 is the dimension of the filter grid.

The spatial Clifford correlation can be defined in a similar manner as the Clifford convolution:

$$\begin{aligned}(\boldsymbol{H} \star_l \boldsymbol{F})(\boldsymbol{x}) &= \int_{E^n} \boldsymbol{H}(\boldsymbol{x}')\boldsymbol{F}(\boldsymbol{x}+\boldsymbol{x}')|d\ x'|,\\ (\boldsymbol{F} \star_r \boldsymbol{H})(\boldsymbol{x}) &= \int_{E^n} \boldsymbol{F}(\boldsymbol{x}+\boldsymbol{x}')\boldsymbol{H}(\boldsymbol{x}')|d\ x'|.\end{aligned} \tag{2.114}$$

This correlation formula can be seen simply as a convolution with a filter that has been reflected with respect to its center.

We can also carry out the scalar Clifford convolution in the Fourier domain. Consider the vector fields $\boldsymbol{f}, \boldsymbol{h} : E^3 \to E^3 \in G_3$. Since

$$\begin{aligned}(\boldsymbol{h} *_s \boldsymbol{f})(\boldsymbol{x}) &= \langle(\boldsymbol{h} *_l \boldsymbol{f})\rangle_0,\\ (\boldsymbol{h} * \boldsymbol{f})_3 &= 0,\end{aligned} \tag{2.115}$$

the Clifford convolution is

$$\mathcal{F}\{(\boldsymbol{h} *_s \boldsymbol{f})\}(\boldsymbol{u}) = \langle\mathcal{F}\{\boldsymbol{h}\}, \mathcal{F}\{\boldsymbol{f}\}\rangle + \langle\mathcal{F}\{\boldsymbol{h}\}, \mathcal{F}\{\boldsymbol{f}\}\rangle_3. \tag{2.116}$$

Since the Clifford Fourier transform of 3D vector fields contains a vector part and a bivector part, the trivector part $\langle\mathcal{F}\{\boldsymbol{h}\}, \mathcal{F}\{\boldsymbol{f}\}\rangle_3$ is generally nonzero.

2.5 Linear Algebra

This section presents the geometric algebra approach to the basic concepts of linear algebra and is presented here for completeness. Next chapter will present related topics in more depth such as differentiation, linear and multilinear functions, eigenblades and tensors formulated in geometric algebra also operators and transformations in common use by mathematicians, physicists, computer scientists, and engineers.

A linear function f maps vectors to vectors in the same space. The extension of f to act linearly on multivectors is possible via the so-called *outermorphism* $\underline{f}$, which defines the action of $\underline{f}$ on r-blades thus:

$$\underline{f}(\boldsymbol{a}_1 \wedge \boldsymbol{a}_2 \wedge \ldots \wedge \boldsymbol{a}_r) = \underline{f}(\boldsymbol{a}_1) \wedge \underline{f}(\boldsymbol{a}_2) \wedge \cdots \wedge \underline{f}(\boldsymbol{a}_r). \tag{2.117}$$

The function $\underline{f}$ is called an outermorphism because $\underline{f}$ preserves the grade of any r-vector it acts upon. The action of $\underline{f}$ on general multivectors is then defined through linearity. The function $\underline{f}$ must therefore satisfy the following conditions:

$$\begin{aligned} \underline{f}(\boldsymbol{a}_1 \wedge \boldsymbol{a}_2) &= \underline{f}(\boldsymbol{a}_1) \wedge \underline{f}(\boldsymbol{a}_2) \\ \underline{f}(A_r) &= \langle \underline{f}(A_r) \rangle_r \\ \underline{f}(\alpha_1 \boldsymbol{a}_1 + \alpha_2 \boldsymbol{a}_2) &= \alpha_1 \underline{f}(\boldsymbol{a}_1) + \alpha_2 \underline{f}(\boldsymbol{a}_2). \end{aligned} \tag{2.118}$$

Accordingly, the outermorphism of a product of two linear functions is the product of the outermorphisms—that is, if $f(\boldsymbol{a}) = f_2(f_1(\boldsymbol{a}))$, we write $\underline{f} = \underline{f}_2 \underline{f}_1$. The *adjoint* $\overline{f}$ of a linear function $\underline{f}$ acting on the vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ can be defined by the property

$$\underline{f}(\boldsymbol{a}) \cdot \boldsymbol{b} = \boldsymbol{a} \cdot \overline{f}(\boldsymbol{b}). \tag{2.119}$$

If $\underline{f} = \overline{f}$, the function is *self-adjoint* and can be represented by a symmetric matrix F ($F = F^T$).

Since the outermorphism preserves grade, the unit pseudoscalar must be mapped onto some multiple of itself; this multiple is called the *determinant* of $\underline{f}$. Thus,

$$\underline{f}(I) = \det(\underline{f}) I. \tag{2.120}$$

This is a particularly simple definition of the determinant, from which many properties of the determinants follow straightforwardly.

2.5.1 Linear Algebra Derivations

In linear algebra, a number of derivations are carried out using frame contraction; instead, we can use vector derivatives as we show in the next equation:

$$\nabla(\boldsymbol{x}\cdot\boldsymbol{v}) = e^i\partial_i(x^i e_j).\boldsymbol{v} = e^i e_j\cdot\boldsymbol{v}\delta^j_i = e^i e_i\cdot\boldsymbol{v} = \boldsymbol{v}. \tag{2.121}$$

This shows that when differentiating a function that depends linearly on $\boldsymbol{x}$, it is simply equivalent to carry out contractions over frame indices. To take advantage of this, one introduces a vector variable $\boldsymbol{x}$ and denotes the derivative with respect to $\boldsymbol{x}$ by ∂_x. In this manner, we can write Eqs. (2.26)–(2.30).

$$\begin{aligned}\partial_x\boldsymbol{x}\cdot\boldsymbol{X}_m &= m\boldsymbol{X}_m,\\ \partial_x\boldsymbol{x}\wedge\boldsymbol{X}_m &= (n-m)\boldsymbol{X}_m,\\ \dot\partial_x\boldsymbol{X}_m\dot{\boldsymbol{x}} &= (-1)^m(n-2m)\boldsymbol{X}_m.\end{aligned} \tag{2.122}$$

The trace of a linear function can be expressed in terms of a vector derivative as follows:

$$Tr(f) = \partial_x\cdot f(\boldsymbol{x}). \tag{2.123}$$

The use of the vector derivatives enables us to write terms of an equation that do not depend on a frame in such a way that reflects this independence. In general, this kind of formulation brings out the intrinsic geometric content of an equation.

2.6 Simplexes

An r-dimensional simplex (r-simplex) in $\mathbb{R}^n$ corresponds to the convex hull of r+1 points, of which at least r are linearly independent. A set of points $\{\boldsymbol{x}_0,\boldsymbol{x}_1,\boldsymbol{x}_2,\ldots,\boldsymbol{x}_r\}$ which define a r-simplex is seen as the *frame* for the r-simplex. One can select $\boldsymbol{x}_0$ as the *base point* or *place* of the simplex. Using the geometric algebra, the related notations for the r-simplex are as follows:

$$\boldsymbol{X}_r \equiv \boldsymbol{x}_0\wedge\boldsymbol{x}_1\wedge\boldsymbol{x}_2\wedge\cdots\wedge\boldsymbol{x}_r = \boldsymbol{x}_0\wedge\bar{\boldsymbol{X}}_r, \tag{2.124}$$

$$\bar{\boldsymbol{X}}_r \equiv (\boldsymbol{x}_1-\boldsymbol{x}_0)\wedge(\boldsymbol{x}_2-\boldsymbol{x}_0)\wedge\cdots\wedge(\boldsymbol{x}_r-\boldsymbol{x}_0), \tag{2.125}$$

where $\bar{\boldsymbol{x}}_i = (\boldsymbol{x}_i-\boldsymbol{x}_0)$ for $i=1,\ldots,r$. $\bar{\boldsymbol{X}}_r$ is called the *tangent* of the simplex, because it is tangent to the r-plane in which the simplex lies, see Fig. 2.5a. The tangent determines the directed distance to the simplex $\bar{\boldsymbol{X}}_r/r!$, and it assigns a definite orientation to the simplex. On the other hand, the *volume* of the simplex is given by $(r!)^{-1}|\bar{\boldsymbol{X}}_r|$ which corresponds to area of the triangle formed by $\bar{\boldsymbol{x}}_1$ and $\bar{\boldsymbol{x}}_2$ in Fig. 2.5b. In general, this is the volume of an $(r+1)$-hedron. Note that $\bar{\boldsymbol{X}}_r$ is independent of the choice of the origin, but $\boldsymbol{X}_r$ is not.

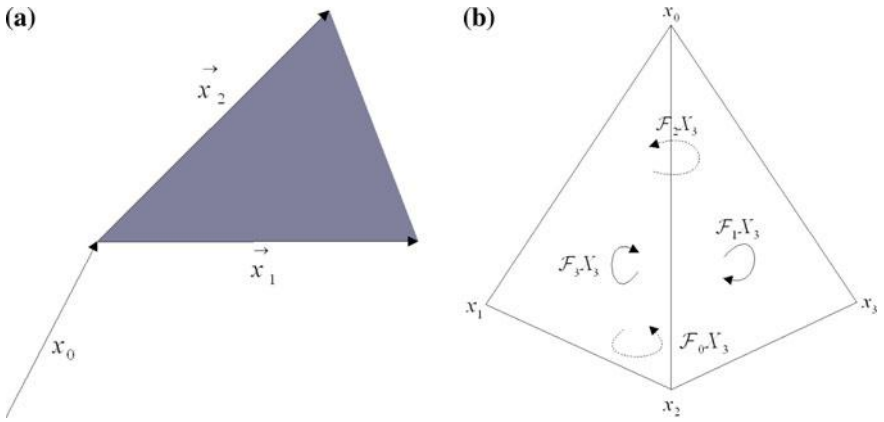


Fig. 2.5 **a** (left) Tangent of the simplex; **b** (right) volume of an 3-hedron

Now, an r-dimensional plane parallel to the subspace of $\boldsymbol{X}_r$ and through the point $\boldsymbol{x}$ is the solution set of

$$(\boldsymbol{y} - \boldsymbol{x}) \wedge \boldsymbol{X}_r = 0, \tag{2.126}$$

for $\boldsymbol{y} \in \mathbb{R}^n$. According to Eq. (2.126), the equation for the plane of the simplex is given by

$$\boldsymbol{y} \wedge \bar{\boldsymbol{X}}_r = \boldsymbol{x}_0 \wedge \bar{\boldsymbol{X}}_r = \boldsymbol{X}_r. \tag{2.127}$$

The term $\boldsymbol{X}_r$ corresponds to the *moment* of the simplex, because it is the wedge product of a point touching the simplex with the orientation entity, think in the moment of a line.

Since Eq. (2.125) represents the pseudoscalar for the simplex frame $\{\boldsymbol{x}_i\}$, it determines also a dual frame $\{\boldsymbol{x}^i\}$ given by Eq. (2.10):

$$\boldsymbol{x}^i \cdot \boldsymbol{x}_j = \delta^i_j. \tag{2.128}$$

The *face* opposite $\boldsymbol{x}_i$ in simplex $\boldsymbol{X}_r$ is described by its moment

$$\mathcal{F}^r_i \boldsymbol{X}_r \equiv \boldsymbol{X}^i_r \equiv \boldsymbol{x}^i \cdot \boldsymbol{X}_r = (-1)^{i+1} \boldsymbol{x}_0 \wedge \boldsymbol{x}_1 \wedge \cdots \wedge \check{\boldsymbol{x}}_i \wedge \cdots \wedge x_r. \tag{2.129}$$

This equation defines a *face* operator $\mathcal{F}_i$, see Fig. 2.5b. Note that the face $\boldsymbol{X}^i_r$ is an $r-1$-dimensional simplex and

$$\boldsymbol{X}_r = \boldsymbol{x}_i \wedge \boldsymbol{X}^i_r, \tag{2.130}$$

for any $0 \leq i \leq r$.

The *boundary* of a simplex $\boldsymbol{X}_r$ is given by the multivector sum as follows:

$$\Omega_b \boldsymbol{X}_r = \sum_{i=0}^{r} \boldsymbol{X}_r^i = \sum_{i=0}^{r} \mathcal{F}_i \boldsymbol{X}_r. \tag{2.131}$$

This boundary operator fulfills

$$\boldsymbol{X}_r = \boldsymbol{x}_i \wedge \Omega_b \boldsymbol{X}_r, \tag{2.132}$$

for any $0 \leq i \leq r$. Taking into account the identity

$$(\boldsymbol{x}_1 - \boldsymbol{x}_0) \wedge (\boldsymbol{x}_2 - \boldsymbol{x}_0) \wedge \cdots \wedge (\boldsymbol{x}_r - \boldsymbol{x}_0) = \sum_{i=0}^{r} (-1)^i \boldsymbol{x}_0 \wedge \cdots \wedge \check{\boldsymbol{x}}_i \wedge \cdots \wedge \boldsymbol{x}_r, \tag{2.133}$$

and Eqs. (2.125) and (2.129), we derive

$$\bar{\boldsymbol{X}}_r = \Omega_b \boldsymbol{X}_r. \tag{2.134}$$

Finally, the following relations hold for the operators $\mathcal{F}_i$ and Ω_b:

$$\mathcal{F}_i \mathcal{F}_i = 0, \tag{2.135}$$
$$\mathcal{F}_i \mathcal{F}_j = -\mathcal{F}_j \mathcal{F}_i, \tag{2.136}$$
$$\mathcal{F}_i \Omega_b = -\Omega_b \mathcal{F}_i, \tag{2.137}$$
$$\Omega_b \Omega_b = 0. \tag{2.138}$$

Note that these operator relations are strongly analogous to relations one find in algebraic topology.

2.7 Exercises

2.1 Given $\boldsymbol{x}, \boldsymbol{y} \in G_{2,0,0}$, expand the bivector $\boldsymbol{x} \wedge \boldsymbol{y}$ in terms of geometric products. Prove that it anticommutes with both $\boldsymbol{x}$ and $\boldsymbol{y}$, however, commutes with any vector outside the plane.

2.2 Prove that the magnitude of bivector $\boldsymbol{x} \wedge \boldsymbol{y}$ is $|\boldsymbol{x}||\boldsymbol{y}| \sin(\theta)$.

2.3 Given $\boldsymbol{a} = e_2 + 2e_1e_2$, $\boldsymbol{b} = e_1 + 2e_2$, and $\boldsymbol{c} = 3e_1e_2 \in G_{2,0,0}$, compute $\boldsymbol{ab}$, $\boldsymbol{ba}$, and $\boldsymbol{bac}$.

2.4 Given the coordinates $(2, 5)$, $(5, 7)$, and $(6, 0)$ for the corners of a triangle, compute in $G_{2,0,0}$, the triangle's area, using multivectors.

2.5 Given $\boldsymbol{a} = e_1 - 2e_2$, $\boldsymbol{b} = e_1 + e_2$, and $\boldsymbol{v} = 5e_1 - e_2 \in G_{2,0,0}$, compute α and β for $\boldsymbol{r} = \alpha\boldsymbol{a} + \beta\boldsymbol{b}$.

2.6 Given $\boldsymbol{a} = 8e_1 - e_2$ and $\boldsymbol{b} = 2e_1 + 2e_2 \in G_{2,0,0}$, compute $\boldsymbol{a}_{||}$ and $\boldsymbol{a}_{\perp}$.

2.7 For each $\boldsymbol{x} \in G_{2,0,0}$, prove $\boldsymbol{x}\bar{\boldsymbol{x}} = \bar{\boldsymbol{x}}\boldsymbol{x}$ and compute the inverse $\boldsymbol{x}^{-1}$ when $\boldsymbol{x}\bar{\boldsymbol{x}} \neq 0$.

2.8 Prove, using multivectors in $G_{2,0,0}$, the following sinus identity:

$$\frac{\sin\alpha}{|a|} = \frac{\sin\beta}{|b|} = \frac{\sin\gamma}{|c|}.$$

2.9 For a Euclidean unit 2-blade, $I^2 = -1$, interpret this geometrically in terms of versors.

2.10 Prove that in $\mathbb{R}^3$ the cross product is equivalent to the following expressions:

$$\boldsymbol{x} \times \boldsymbol{y} = -I\boldsymbol{x} \wedge \boldsymbol{y} = -\boldsymbol{x} \cdot (I\boldsymbol{y}) = \boldsymbol{y} \cdot (I\boldsymbol{x}).$$

2.11 Interpret geometrically the equations of Exercise 1.9 and establish that the following expressions are true:

$$\boldsymbol{x} \times (\boldsymbol{y} \times \boldsymbol{z}) = -\boldsymbol{x} \cdot (\boldsymbol{y} \wedge \boldsymbol{z}) = -(\boldsymbol{x} \cdot \boldsymbol{y}\boldsymbol{z} - \boldsymbol{x} \cdot \boldsymbol{z}\boldsymbol{y}),$$

and

$$\boldsymbol{x} \cdot (\boldsymbol{y} \times \boldsymbol{z}) = [\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}] = \boldsymbol{x} \wedge \boldsymbol{y} \wedge \boldsymbol{z}I^{-1}.$$

2.12 Given the 2-blade (bivector) $\boldsymbol{X} = e_1 \wedge (e_2 - e_3)$ that represents a plane, check if the following vectors lie in that plane: (i) e_1; (ii) $e_1 + e_2$; (iii) $e_1 + e_2 + e_3$, and (iv) $e_1 - 2e_2 + e_3$.

2.13 In 3-D space, a trivector $\boldsymbol{x} \wedge \boldsymbol{y} \wedge \boldsymbol{z}$ can be written in terms of a determinant:

$$\boldsymbol{x} \wedge \boldsymbol{y} \wedge \boldsymbol{z} = det([\boldsymbol{x}\boldsymbol{y}\boldsymbol{z}])e_1 \wedge e_2 \wedge e_3, \tag{2.139}$$

where $[\boldsymbol{x}\boldsymbol{y}\boldsymbol{z}]$ is a matrix with column vectors. Express the trivector not with a matrix but fully in terms of geometric algebra.

2.14 Show in $G_{3,0,0}$ that the pseudoscalar I commutes with e_1, e_2, and e_3. Compute the volume of a parallelepiped spanned by the vectors $\boldsymbol{a} = 3e_1 - 4e_2 + 4e_3$, $\boldsymbol{b} = 2e_1 + 4e_2 - 2e_3$, and $\boldsymbol{c} = 4e_1 - 2e_2 - 3e_3$.

2.15 Compute $\boldsymbol{A}\boldsymbol{B}$ in $G_{3,0,0}$ if $\boldsymbol{A} = 3e_1e_2 + 6e_2e_3$ and $\boldsymbol{B} = \boldsymbol{a} \wedge \boldsymbol{b}$ where $\boldsymbol{a} = 3e_1 - 4e_2 + 5e_3$ and $\boldsymbol{b} = 2e_1 + 4e_2 - 2e_3$.

2.16 Given in $G_{3,0,0}$ the vector $\boldsymbol{a} = 3e_1 + 5e_2 + 6e_3$ and the bivector $\boldsymbol{B} = 3e_2e_3 + 6e_3e_1 + 2e_1e_2$, compute the parallel and orthogonal projections of $\boldsymbol{a}$ with respect to the plane $\boldsymbol{B}$. Then compute the cross product of these projected components and interpret geometrically the dual relationship of the result with respect to $\boldsymbol{B}$.

2.17 Show in $G_{3,0,0}$ that the geometric product of the bivector $\boldsymbol{A}$ and any vector $\boldsymbol{x}$ can be decomposed as follows:

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{A} \cdot \boldsymbol{x} + \frac{1}{2}(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{x}\boldsymbol{A}) + \boldsymbol{A} \wedge \boldsymbol{x}.$$

2.18 Given $\boldsymbol{x} = 1 + \boldsymbol{a} + \boldsymbol{B}$, where $\boldsymbol{a} \in \mathbb{R}^3$ and $\boldsymbol{B} \in \bigwedge^2 \mathbb{R}^3$.

(a) The outer inverse of $\boldsymbol{x}$ is $\boldsymbol{x}^{\wedge(-1)} = 1 - \boldsymbol{a} - \boldsymbol{B} + \alpha \boldsymbol{a} \wedge \boldsymbol{B}$, where $\alpha \in \mathbb{R}$. Compute α (*Hint*: Use the power series or $\boldsymbol{x} \wedge \boldsymbol{x}^{\wedge(-1)} = 1$;).

(b) The outer square root of $\boldsymbol{x}$ is $\boldsymbol{x}^{\wedge(\frac{1}{2})} = 1 + \frac{1}{2}\boldsymbol{a} + \frac{1}{2}\boldsymbol{B} + \beta \boldsymbol{a} \wedge \boldsymbol{B}$, where $\beta \in \mathbb{R}$. Compute β (*Hint*: $\boldsymbol{x}^{\wedge(\frac{1}{2})} \wedge \boldsymbol{x}^{\wedge(\frac{1}{2})} = \boldsymbol{x}$.).

(c) Give a geometric interpretation of the outer inverse and of the outer square root and suggest their applications.

2.19 In 4-D space with an associated orthonormal basis $\{e_i\}_{i=1}^4$, project the 2-blade $\boldsymbol{Y} = (e_1 + e_2) \wedge (e_3 + e_4)$ onto the 2-blade $\boldsymbol{X} = e_1 \wedge e_3$. Compute then the rejection as the difference of $\boldsymbol{Y}$ and its projection. Prove that this is not a blade.

2.20 In $\mathbb{R}^4$, show that the 2-vector $\boldsymbol{X} = e_1 \wedge e_2 + e_3 \wedge e_4$ is not a 2-blade which means that it cannot be expressed as the wedge product of two vectors. Hint: Given $\boldsymbol{X} = \boldsymbol{x} \wedge \boldsymbol{y}$, express $\boldsymbol{x}$ and $\boldsymbol{y}$ in terms of the basis vectors, expand the wedge product, and try to solve the resulting scalar equations.

2.21 In Exercise 2.20, check that the 2-vector $\boldsymbol{X} = e_1 \wedge e_2 + e_3 \wedge e_4$ does not contain any vector other than 0. We mean by *contain*, denoted $\boldsymbol{X} \subseteq \boldsymbol{Y}$, the case when all vectors in $\boldsymbol{X}$ are also in $\boldsymbol{Y}$.

2.22 In $G_{2,0,0}$, the multivectors $\boldsymbol{X}$ and $\boldsymbol{Y}$ are given by

$$\boldsymbol{X} = x_0 + x_1 e_1 + x_2 e_2 + x_3 e_1 e_2 \qquad \boldsymbol{Y} = y_0 + y_1 e_1 + y_2 e_2 + y_3 e_1 e_2,$$

where the basis vectors e_1, e_2 are orthonormals. Compute their geometric product

$$\boldsymbol{X}\boldsymbol{Y} = w_0 + w_1 e_1 + w_2 e_2 + w_3 e_1 e_2,$$

making explicit w_0, w_1, w_2, w_3 and establish that $\langle \boldsymbol{X}\boldsymbol{Y} \rangle = \langle \boldsymbol{Y}\boldsymbol{X} \rangle$.

2.23 Expand out the trivector $\boldsymbol{x} \wedge (\boldsymbol{y} \wedge \boldsymbol{z})$ in terms of geometric products. Is the result antisymmetric on $\boldsymbol{x}$ and $\boldsymbol{y}$? Justify why one can also write the following equality:

$$\boldsymbol{x} \wedge \boldsymbol{y} \wedge \boldsymbol{z} = \frac{1}{2}(\boldsymbol{x}\boldsymbol{y}\boldsymbol{z} - \boldsymbol{w} \wedge \boldsymbol{y} \wedge \boldsymbol{x}),$$

and prove that the following equation is true:

$$\boldsymbol{x} \wedge \boldsymbol{y} \wedge \boldsymbol{z} = \frac{1}{6}(\boldsymbol{x}\boldsymbol{y}\boldsymbol{z} + \boldsymbol{z}\boldsymbol{x}\boldsymbol{y} + \boldsymbol{y}\boldsymbol{w}\boldsymbol{x} - \boldsymbol{x}\boldsymbol{z}\boldsymbol{y} - \boldsymbol{y}\boldsymbol{a}\boldsymbol{z} - \boldsymbol{w}\boldsymbol{y}\boldsymbol{x}).$$

2.24 Prove using wedge and inner products the formula $\boldsymbol{x} \times (\boldsymbol{y} \times z) = \boldsymbol{x} \cdot (z\boldsymbol{y}) - \boldsymbol{x} \cdot (\boldsymbol{y}z)$.
Using wedge and inner products of the vector calculus formula.

2.25 Compute the area of a parallelogram spanned by the vectors $\boldsymbol{x} = 2e_1 + e_3$ and $\boldsymbol{y} = e1 - e_3$ relative to the area of $e_1 \wedge e_3$.

2.26 In $G_{3,0,0}$, compute the intersection of the non-homogeneous line L_1 with position vector e_3 and direction $e_2 + e_3$ and the line L_2 with position vector $e_2 - e_3$ and direction e_2.

2.27 The projection of a vector $\boldsymbol{v}$ with respect to a plane $\boldsymbol{A}_r$ is given by

$$\boldsymbol{v} = \boldsymbol{v}_{\parallel} + \boldsymbol{v}_{\perp} = P^{\parallel}_{\boldsymbol{A}_r} + P^{\perp}_{\boldsymbol{A}_r},$$

justify why the vector $\boldsymbol{v} \cdot \boldsymbol{A}_r$ is orthogonal to the plane formed by $\boldsymbol{v}_{\parallel}$ and $\boldsymbol{v}_{\perp}$.

2.28 With a bivector $\boldsymbol{B}$ and the general multivectors $\boldsymbol{X}$, $\boldsymbol{Y}$, prove that

$$\boldsymbol{B} \times (\boldsymbol{X}\boldsymbol{Y}) = (\boldsymbol{B} \times \boldsymbol{X})\boldsymbol{Y} + \boldsymbol{X}(\boldsymbol{B} \times \boldsymbol{Y}),$$

hence that

$$\boldsymbol{B} \times (\boldsymbol{x} \wedge \boldsymbol{X}_r) = (\boldsymbol{B} \cdot \boldsymbol{x}) \wedge \boldsymbol{X}_r + \boldsymbol{x} \wedge (\boldsymbol{B} \times \boldsymbol{X}_r).$$

Use these results to establish that the operation of commuting with a bivector is grade preserving.

2.29 Affine transformations: consider the standard orthonormal basis of $\mathbb{R}^{n,0}$ for the shear transformation $g_s : \boldsymbol{x} \to g_s(\boldsymbol{x}) = \boldsymbol{x} + s(\boldsymbol{x} \cdot e_2)e_1$; compute the transformation matrices $[g_s]$ and $[g_s^*]$ both to act on vectors. Depict your results to see the shear effect of a planar line and its normal vector.

2.30 Simplex: given the three points $\boldsymbol{x}_0 = e_3, \boldsymbol{x}_1 = e_2 + e_3, \boldsymbol{x}_2 = e_1 + e_3$, compute using CLICAL [13], CLUCAL [14], or eClifford [15], the simplex tangent $\bar{\boldsymbol{X}}_2$, the simplex volume $(2!)^{-1}|\bar{\boldsymbol{X}}_2|$, and the simplex moment $\boldsymbol{X}_2 = \boldsymbol{x}_0 \wedge \bar{\boldsymbol{X}}_2$. Show that if $\boldsymbol{x}_0 = 0$ then the simplex passes through the origin and its moment vanishes.

2.31 Simplex: given the four points $\boldsymbol{x}_0 = e_1, \boldsymbol{x}_1 = e_2, \boldsymbol{x}_2 = e_1 + e_2$, and $\boldsymbol{x}_3 = e_3$, compute using CLICAL [13], CLUCAL [14], or eClifford [15], the four faces $\mathcal{F}_i^3 \boldsymbol{X}_3$ opposite to the points $\boldsymbol{x}_i$ $i = 0, \ldots, 3$.

2.32 Using the points given in 2.31, compute using CLICAL [13], CLUCAL [14], or eClifford [15], the boundary $\Omega_b X_3$ of the simplex $\boldsymbol{X}_3$.

2.33 Using the points given in Exercise 1.24, check using CLICAL [13], CLUCAL [14], or eClifford [15], if the following relations hold for the operators $\mathcal{F}_i$ and Ω_b:

$$\begin{aligned}
\mathcal{F}_i \mathcal{F}_i &= 0, \\
\mathcal{F}_i \mathcal{F}_j &= -\mathcal{F}_j \mathcal{F}_i, \\
\mathcal{F}_i \Omega_b &= -\Omega_b \mathcal{F}_i, \\
\Omega_b \Omega_b &= 0.
\end{aligned}$$

2.34 Using CLICAL [13], CLUCAL [14], or eClifford [15] compute the join $\boldsymbol{X} \cup \boldsymbol{Y}$ and meet $\boldsymbol{X} \cap \boldsymbol{Y}$ for the following blades: (i) $\boldsymbol{X} = e_2$ and $\boldsymbol{Y} = 3e_1$; (ii) $\boldsymbol{X} = e_1$ and $\boldsymbol{Y} = 4e_1$; (iii) $\boldsymbol{X} = e_1$ and $\boldsymbol{Y} = -e_2$; (iv) $\boldsymbol{X} = e_1 \wedge e_2$ and $\boldsymbol{Y} = 2e_1$; (v) $\boldsymbol{X} = (e_1 + e_2)/\sqrt{2}$ and $\boldsymbol{Y} = e_1$; (vi) $\boldsymbol{X} = e_1 \wedge e_2$ and $\boldsymbol{Y} = 0.00001e_1 + e_2$, and (vii) $\boldsymbol{X} = e_1 \wedge e_2$ and $\boldsymbol{Y} = \cos(\theta)e_1 + \sin(\theta)e_2$.

References

1. Needham, T. (2003). *Visual analysis, complex*. New York: Oxford University Press. Reprinted.
2. Li, H., Hestenes, D., & Rockwood, A. (2001). Generalized homogeneous coordinates for computational geometry. In G. Sommer (Ed.), *Geometric computing with Clifford algebra* (pp. 27–59). New York: Springer-Verlag.
3. Grassmann, H. (1877). Der Ort der Hamilton'schen Quaternionen in der Ausdehnungslehre. *Mathematische Annalen*, *12*, 375.
4. Clifford, W. K. (1878). Applications of Grassmann's extensive algebra. *American Journal of Mathematics*, *1*, 350–358.
5. Hestenes, D. (1966). *Space-time algebra*. London: Gordon and Breach.
6. Hestenes, D., & Sobczyk, G. (1984). *Clifford algebra to geometric calculus: A unified language for mathematics and physics*. Dordrecht: D. Reidel.
7. Hestenes, D. (1986). *New foundations for classical mechanics*. Dordrecht: D. Reidel.
8. Hestenes, D., & Ziegler, R. (1991). Projective geometry with Clifford algebra. *Acta Applicandae Mathematicae*, *23*, 25–63.
9. Bayro-Corrochano, E. (2018). *Geometric algebra applications vol. I computer vision, graphics and neurocomputing*. Springer Verlag.
10. Doran, C., & Lasenby, A. (2005). *Geometric algebra for physicists*. Cambridge University Press.
11. Lounesto, P. (1997). *Clifford algebras and spinors*. Cambridge: Cambridge University Press.
12. Hestenes, D. (1991). The design of linear algebra and geometry. *Acta Applicandae Mathematicae*, *23*, 65–93.
13. Lounesto, P. (1987). CLICAL software packet and user manual. Helsinki University of Technology of Mathematics, Research report A248.
14. Perwass, C. B. U. (2006). CLUCal. http://www.clucal.info/.
15. Ablamowicks R. eCLIFFORD Software packet using Maple for Clifford algebra. Computations. http://math.tntech.edu/rafal.

Chapter 3
Lie Algebras, Lie Groups, and Algebra of Incidence

3.1 Introduction

We have learned that readers of the work of Hestenes and Sobzyk [1, Chap. 8] and a late article of Doran et al. [2] section IV may have difficulties to understand the subject, and practitioners have difficulties to try the equations in certain applications. For this reason, this chapter reviews concepts and equations most of them introduced by Hestenes and Sobzyk [1, Chap. 8] and the article of Doran et al. [2, Sect. IV]. This chapter is written in a clear manner for readers interested in applications in computer science and engineering. The explained equations will be required to understand advanced applications in next chapters.

In this chapter, we give the fundamentals of Lie algebra and Lie groups using the computational frameworks of the null cone and the n-dimensional affine plane. Using Lie algebra within this computational framework has the advantage that it is easily accessible to the reader because there is a direct translation of the familiar matrix representations to representations using bivectors from geometric algebra. Generally speaking, Lie group theory is the appropriate tool for the study and analysis of the action of a group on a manifold. Since we are interested in purely geometric computations, the use of geometric algebra is appropriate for carrying out computations in a coordinate-free manner by using a bivector representation of the most important Lie algebras. We will represent Lie operators using bivectors for the computation of a variety of invariants. This section begins with a brief introduction into the Lie group theory. We study the $Pin(n)$ and $Spin(n)$ groups, the projective transformations using the null geometric algebra over $\mathbb{R}^{3,3}$.

3.2 Lie Algebras and Bivector Algebras

In an abstract way, a Lie group is defined as a manifold $\mathcal{M}$ together with a product $\zeta(x, y)$. Points on the manifold can be labeled as vectors which can be seen as lying in a higher dimensional embedding space. The product $\zeta(x, y)$ encodes the group

© Springer Nature Switzerland AG 2020

E. Bayro-Corrochano, *Geometric Algebra Applications Vol. II*,
https://doi.org/10.1007/978-3-030-34978-3_3

product; it takes two points as arguments and returns a third one. The following conditions applied to the product $\zeta(x, y)$ ensure that this product has the correct group properties.

i. Closure: $\zeta(x, y) \in \mathcal{M}, \forall\, x, y \in \mathcal{M}$.
ii. Identity: There exists en element $e \in \mathcal{M}$ such that $\zeta(e, x) = \zeta(x, e) = x, \quad \forall\, x \in \mathcal{M}$.
iii. Inverse: For every element $x \in \mathcal{M}$ there exists a unique element $\hat{x}$ such that $\zeta(x, \hat{x}) = \zeta(\hat{x}, x) = e$.
iv. Associativity. $\zeta[\zeta(x, y), z] = \zeta[x, \zeta(y, z)], \forall\, x, y, z \in \mathcal{M}$.

In general, any manifold equipped with a product which fulfills the properties $i - iv$ is called a Lie group manifold. Most of the group properties can be elucidated by analyzing these properties near to the identity element e. The product $\zeta(x, y)$ induces a $Lie\, bracket$ structure on the elements of the tangent space at the identity e. This tangent space is linear, and it is spanned by a set of Lie algebra vectors also called Lie algebra generators. These basis vectors together with their bracket form a Lie algebra.

3.3 Lie Groups and Lie Algebras of Rotors

In the geometric algebra framework, we can reveal conveniently the properties of Lie algebra using bivector algebras, and furthermore bivector algebra makes possible to represent Lie groups of the general linear group using few coefficients, e.g., a 3D rotation matrix has nine real entries and its isomorph counterpart a rotor only four.

3.3.1 Lie Group of Rotors

This subsection is based on [3, 11.3.2–11.3.5]. Let us analyze the Lie group properties of the 3D geometric algebra G_3 using rotors. Choose a rotor $\boldsymbol{R}_\alpha$ and imagine a family of rotors, $\boldsymbol{R}(\alpha)$, for which

$$\boldsymbol{R}(0) = 1, \qquad \boldsymbol{R}(\alpha) = \boldsymbol{R}_\alpha. \tag{3.1}$$

This shows that the rotor can be obtained from the identity by a continuous set of rotor transformations. In fact, there are many possible paths to connect $\boldsymbol{R}_\alpha$ to the identity, but there is only one path which has the additional property that

$$\boldsymbol{R}(\alpha + \beta) = \boldsymbol{R}(\alpha)\boldsymbol{R}(\beta). \tag{3.2}$$

These properties belong to the one-parameter subgroup of the rotor group, and it represents to all rotations in a fixed oriented plane. Now consider a family of vectors

$\boldsymbol{v}(\alpha) = \boldsymbol{R}\boldsymbol{v}_0\tilde{\boldsymbol{R}}$, where $\boldsymbol{v}_0$ is some fixed initial vector, and the differentiation of the relationship $\boldsymbol{R}\tilde{\boldsymbol{R}} = 1$

$$\frac{d}{d\alpha}(\boldsymbol{R}\tilde{\boldsymbol{R}}) = \boldsymbol{R}'\tilde{\boldsymbol{R}} + \boldsymbol{R}\tilde{\boldsymbol{R}}' = 0, \tag{3.3}$$

where the dash symbol stands for the differentiation. By using these relations, we can compute

$$\begin{aligned}\frac{d}{d\alpha}\boldsymbol{v}(\alpha) &= \boldsymbol{R}'\boldsymbol{v}_0\tilde{\boldsymbol{R}} + \boldsymbol{R}\boldsymbol{v}_0\tilde{\boldsymbol{R}}' = (\boldsymbol{R}'\tilde{\boldsymbol{R}})\boldsymbol{v}(\alpha) - \boldsymbol{v}(\alpha)(\boldsymbol{R}'\tilde{\boldsymbol{R}}),\\ &= (-2\boldsymbol{R}'\tilde{\boldsymbol{R}})\cdot\boldsymbol{v}(\alpha).\end{aligned} \tag{3.4}$$

Here we make use of the inner product between a bivector and a vector. In this equation, the derivative of a vector should yield a vector as well, as given by the inner product of $(-2\boldsymbol{R}'\tilde{\boldsymbol{R}})\cdot\boldsymbol{v}(\alpha)$. Since $\boldsymbol{R}'\tilde{\boldsymbol{R}}$ is a bivector now called $\boldsymbol{B}(\alpha)$

$$-2\boldsymbol{R}'\tilde{\boldsymbol{R}} = \boldsymbol{B}(\alpha) \longrightarrow \frac{d}{d\alpha}(\boldsymbol{R}) = \boldsymbol{R}' = -\frac{1}{2}\boldsymbol{B}(\alpha)\boldsymbol{R}. \tag{3.5}$$

This result is valid for any parameterized set of rotors; however, restricted to the curve defined by Eq. (3.2), one gets the following equation:

$$\begin{aligned}\frac{d}{d\alpha}\boldsymbol{R}(\alpha+\beta) &= -\frac{1}{2}\boldsymbol{B}(\alpha+\beta)\boldsymbol{R}(\alpha+\beta) = -\frac{1}{2}\boldsymbol{B}(\alpha+\beta)\boldsymbol{R}(\alpha)\boldsymbol{R}(\beta),\\ &= \frac{d}{d\alpha}[\boldsymbol{R}(\alpha)\boldsymbol{R}(\beta)] = -\frac{1}{2}\boldsymbol{B}(\alpha)\boldsymbol{R}(\alpha)\boldsymbol{R}(\beta).\end{aligned} \tag{3.6}$$

This result shows that the bivector $\boldsymbol{B}$ follows constant along this curve. Integrating Eq. (3.5), one gets

$$\boldsymbol{R}(\alpha) = e^{-\frac{\alpha\boldsymbol{B}}{2}}. \tag{3.7}$$

Let us find the equivalent bivector expression of a rotated vector

$$\boldsymbol{v}(\alpha) = \boldsymbol{R}(\alpha)\boldsymbol{v}_0\tilde{\boldsymbol{R}}(\alpha) = e^{-\frac{\alpha\boldsymbol{B}}{2}}\boldsymbol{v}_0 e^{\frac{\alpha\boldsymbol{B}}{2}}. \tag{3.8}$$

Differentiating it successively,

$$\begin{aligned}\frac{d\boldsymbol{v}}{d\alpha} &= e^{-\frac{\alpha\boldsymbol{B}}{2}}\boldsymbol{v}_0\cdot\boldsymbol{B}e^{\frac{\alpha\boldsymbol{B}}{2}},\\ \frac{d^2\boldsymbol{v}}{d\alpha^2} &= e^{-\frac{\alpha\boldsymbol{B}}{2}}(\boldsymbol{v}_0\cdot\boldsymbol{B})\cdot\boldsymbol{B}e^{\frac{\alpha\boldsymbol{B}}{2}},\\ &\cdots\\ &\texttt{etc.}\end{aligned} \tag{3.9}$$

One sees that for every derivative, the rotated vector is doted with one extra $\boldsymbol{B}$. Since this operation is grade preserving, the resulting rotated vector can be expressed as a useful Taylor expansion, c.f. [3, 11.65],

$$\boldsymbol{R}(\alpha)\boldsymbol{v}\tilde{\boldsymbol{R}}(\alpha) = e^{-\frac{\alpha \boldsymbol{B}}{2}}\boldsymbol{v}e^{\frac{\alpha \boldsymbol{B}}{2}} = \boldsymbol{v} + \boldsymbol{v}\cdot\boldsymbol{B} + \frac{1}{2|}(\boldsymbol{v}\cdot\boldsymbol{B})\cdot\boldsymbol{B} + \cdots \tag{3.10}$$

3.3.2 Bivector Algebra

It is easy to prove that the operation of commuting a multivector with a bivector is always grade preserving. In particular, the commutator of two bivectors yields a third bivector; thus, it follows that the space of bivectors is closed under the commutator product. This closed algebra is in fact a Lie algebra which preserves most of the properties of the associated Lie group of rotors. Thus, the Lie group of rotors is formed from the bivector algebra by means of the exponentiation operation.

In Lie algebra, elements are acted on by applying the Lie bracket, which is anti-symmetric and satisfies the so-called *Jacobi identity*. In bivector algebra, the Lie bracket is just the commutator of bivectors. Given three bivectors $\boldsymbol{X}$, $\boldsymbol{Y}$, $\boldsymbol{Z}$, the Jacobi identity is given by

$$(\boldsymbol{X}\times\boldsymbol{Y})\times\boldsymbol{Z} + (\boldsymbol{Z}\times\boldsymbol{X})\times\boldsymbol{Y} + (\boldsymbol{Y}\times\boldsymbol{Z})\times\boldsymbol{X} = 0. \tag{3.11}$$

This equation is simple and it requires the expansion of each bivector cross product.

Given a set of basis bivectors $\{\boldsymbol{B}_i\}$, the commutator of any pair of these bivectors returns a third bivector which can be expanded as a linear combination of this basis bivectors. Therefore, we can express the commutator operation as follows:

$$\boldsymbol{B}_j \times \boldsymbol{B}_k = C^i_{jk}\boldsymbol{B}_i, \tag{3.12}$$

the set if C^i_{jk} is called the *structure constants* of the Lie algebra and it can be used to recover most of properties of the corresponding Lie group. The classification of all possible Lie algebras can be carried out using the structure constants. The mathematician E. Cartan completed the solution of this problem.

Let us define the *Killing form* of Lie algebra. First of all, the *adjoint* representation of a Lie group is defined entirely in terms of certain functions which maps the Lie algebra onto itself. Each element of Lie group induces an adjoint representation, due its action, on the Lie algebra. Let us illustrate this using the Lie group of the rotor groups G^+_3. Given the bivectors $\boldsymbol{U}$, $\boldsymbol{V}$, the map of rotor group is

$$\boldsymbol{V} \to \boldsymbol{R}\boldsymbol{V}\tilde{\boldsymbol{R}} = Ad_R(\boldsymbol{V}). \tag{3.13}$$

This representation fulfills for consecutive rotations

$$Ad_{R_1}(Ad_{R_2}(\boldsymbol{V})) = Ad_{R_1 R_2}(\boldsymbol{V}). \tag{3.14}$$

Now, an adjoint of the rotor group representation induces an adjoint representation $ad_{\frac{U}{2}}$ of the Lie algebra as follows, c.f. [3, 11.82]:

$$ad_{\frac{U}{2}}(\boldsymbol{V}) = \boldsymbol{U} \times \boldsymbol{V}. \tag{3.15}$$

The adjoint representation of an element of the Lie algebra can be seen as linear map in the space of bivectors. Note that the matrix corresponding to the adjoint representation of the bivector $\boldsymbol{V}_j$ is determined by the structure coefficients, namely, c.f. [3, 11.83],

$$(ad_{V_j})^i_k = 2C^i_{jk}. \tag{3.16}$$

Finally, the Killing form for a Lie algebra is determined in terms of the adjoint representation

$$\mathcal{K}(\boldsymbol{U}, \boldsymbol{V}) = tr(ad_U ad_V), \tag{3.17}$$

up to a normalization. Now, the Killing form for a bivector algebra is simply computed using the inner product, c.f. [3, 11.85], i.e.,

$$\mathcal{K}(\boldsymbol{U}, \boldsymbol{V}) = \boldsymbol{U} \cdot \boldsymbol{V}. \tag{3.18}$$

According to Eq. (3.18), the rotor groups in Euclidean space have a negative-definite Killing form. A Lie algebra with negative-definite Killing form is called to be of compact type, and consequently the corresponding Lie group is compact.

3.4 Complex Structures and Unitary Groups

In geometric algebra, one can represent complex numbers defining one basis vector for the real axis and other which squares to -1 for the imaginary axis. This suggests us that a n-D complex space could have a natural realization in a $2n$-D space. In this section, we will explain the Lie algebras and Lie groups of the rotors and discuss the role of the *doubling bivector* $\boldsymbol{J}$ in $2n$-D spaces by the unitary groups. This section is based on [3, Sect. 11.4.1].

3.4.1 The Doubling Bivector

Suppose that the n-D space has some arbitrary vector basis $\{e_i\}$, which need not to be orthonormal. We introduce a second set of basis vectors $\{f_i\}$ perpendicular to $\{e_i\}$ which fulfill the following properties:

$$e_i \cdot e_j = f_i \cdot f_j = \delta_{ij} \qquad e_i \cdot f_j = 0, \qquad \forall i, j = 1, \ldots, n. \tag{3.19}$$

One introduces a complex structure through the so-called *doubling bivector*, c.f. [3, 11.87],

$$\boldsymbol{J} = e_1 \wedge f^1 + e_2 \wedge f^2 + e_3 \wedge f^3 + \cdots + e_n \wedge f^n = e_i \wedge f^i, \tag{3.20}$$

where $\{f^i\}$ are the reciprocal vectors to the frame $\{f_i\}$ frame. For this and next equations, we assume that repeated indices are summed from $1, \ldots, n$. Let us show that the bivector $\boldsymbol{J}$ is independent of the initial choice of the frame $\{e_i\}$. Introduce a second pair of frames $\{e'_i\}$ and $\{f'_i\}$ to compute a bivector $\boldsymbol{J}'$ as follows:

$$\boldsymbol{J}' = e'_i \wedge f'^i = e'_i \cdot e^j e_j \wedge f'^i = f'_i \cdot f^j e_j \wedge f'^i = e_j \wedge f^j = \boldsymbol{J}. \tag{3.21}$$

In the case, if the $\{e_i\}$ frame is chosen to be orthonormal, we obtain

$$\boldsymbol{J} = e_1 f_1 + e_2 f_2 + e_3 f_3 + \cdots + e_n f_n = e_i f_i = \boldsymbol{J}_1 + \boldsymbol{J}_2 + \cdots + \boldsymbol{J}_n. \tag{3.22}$$

This sum consists of n commuting blades of grade two; each bivector $\boldsymbol{J}_i$ plays the role of an imaginary number representing an oriented i-th rotation plane. The doubling bivector satisfies the following conditions:

$$\begin{aligned} \boldsymbol{J} \cdot f_i &= (e_j \wedge f_j) \cdot f_i = e_j \delta_{ij} = e_i, \\ \boldsymbol{J} \cdot e_i &= (e_j \wedge f_j) \cdot e_i = -f_i. \end{aligned} \tag{3.23}$$

This computation shows the interesting role of the doubling vector $\boldsymbol{J}$ which relates one n-D space with the other. Other useful relations follow

$$\begin{aligned} \boldsymbol{J} \cdot (\boldsymbol{J} \cdot e_i) &= -\boldsymbol{J} \cdot f_i = -e_i, \\ \boldsymbol{J} \cdot (\boldsymbol{J} \cdot f_i) &= \boldsymbol{J} \cdot e_i = -f_i, \end{aligned} \tag{3.24}$$

thus for any vector $\boldsymbol{v}$ in the 2n-D space

$$\boldsymbol{J} \cdot (\boldsymbol{J} \cdot \boldsymbol{v}) = (\boldsymbol{v} \cdot \boldsymbol{J}) \cdot \boldsymbol{J} = -\boldsymbol{v}, \qquad \forall\, \boldsymbol{v}. \tag{3.25}$$

Similar as in Eq. (3.10), the Taylor expansion involving the doubling vector $\boldsymbol{J}$ is the description of a series of coupled rotations with respect to $e_i \wedge f_i$ planes. Using

trigonometric expressions, the Taylor expansion of $\boldsymbol{R}\boldsymbol{v}\tilde{\boldsymbol{R}}$ can be written as follows, c.f. [3, 11.94]:

$$\begin{aligned}\boldsymbol{R}\boldsymbol{v}\tilde{\boldsymbol{R}} = e^{-\boldsymbol{J}\frac{\theta}{2}}\boldsymbol{v}e^{\boldsymbol{J}\frac{\theta}{2}} &= \boldsymbol{v} + \theta\boldsymbol{v}\cdot\boldsymbol{J} + \frac{\theta^2}{2!}(\boldsymbol{v}\cdot\boldsymbol{J})\cdot\boldsymbol{J}\cdots \\ &= \left(1 - \frac{\theta^2}{2!} + \frac{\theta^4}{4!} + \cdots\right)\boldsymbol{v} + \left(\theta - \frac{\theta^3}{3!} - \cdots\right)\boldsymbol{v}\cdot\boldsymbol{J} \\ &= \cos\theta\boldsymbol{v} + \sin\theta\boldsymbol{v}\cdot\boldsymbol{J}. \end{aligned} \tag{3.26}$$

3.4.2 Unitary Groups $U(n)$ and Unitary Lie Algebras $u(n)$

This section is based on [3, Sects. 11.4.1, 11.4.2]. In the study of unitary groups, one focuses on the Hermitian inner product, as it is left invariant under group actions. Consider a pair of complex vectors

$$z_{1i} = x_i + iy_i, \quad z_{2i} = z_i + iw_i \in \mathbb{C}. \tag{3.27}$$

Their Hermitian inner product is

$$\langle z_1|z_2\rangle = z_1{}_i^* z_2{}^i = x_i z_i + y_i w_i + i(y_i z_i - x_i w_i). \tag{3.28}$$

We are looking for an analog for the 2n-D space. A real vector $\boldsymbol{x}$ in the $2n$-dimensional algebra can be mapped to a set of complex coefficients $\{x_i\}$. For that, let us first introduce the following vectors:

$$x_i = \boldsymbol{x}\cdot e_i + i\boldsymbol{x}\cdot f_i, \qquad y_i = \boldsymbol{z}\cdot e_i + i\boldsymbol{w}\cdot f_i. \tag{3.29}$$

The complex inner product reads

$$\begin{aligned}\langle\boldsymbol{x}|\boldsymbol{y}\rangle &= x^i y_i^* = (\boldsymbol{x}\cdot e^i + i\boldsymbol{x}\cdot f^i)(\boldsymbol{z}\cdot e_i + \boldsymbol{w}\cdot f_i), \qquad (3.30)\\ &= \boldsymbol{x}\cdot e^i\,\boldsymbol{y}\cdot e_i + \boldsymbol{x}\cdot f^i\,\boldsymbol{y}\cdot f_i + i(\boldsymbol{x}\cdot f^i\,\boldsymbol{y}\cdot e_i - \boldsymbol{x}\cdot e^i\,\boldsymbol{y}\cdot f_i), \\ &= \boldsymbol{x}\cdot\boldsymbol{y} + i(\boldsymbol{x}\wedge\boldsymbol{y})\cdot\boldsymbol{J}. \qquad (3.31)\end{aligned}$$

Here, $\boldsymbol{x}\cdot\boldsymbol{y}$ is the real part, and the imaginary part is an antisymmetric product formed by projecting the bivector $(\boldsymbol{x}\wedge\boldsymbol{y})$ onto $\boldsymbol{J}$.

In order to prove the invariance group of the Hermitian inner product,

$$(\boldsymbol{x}'\wedge\boldsymbol{y}')\cdot\boldsymbol{J} = (\boldsymbol{x}\wedge\boldsymbol{y})\cdot\boldsymbol{J}, \tag{3.32}$$

where $\boldsymbol{x}' = \boldsymbol{R}\boldsymbol{x}\tilde{\boldsymbol{R}}$ and $\boldsymbol{y}' = \boldsymbol{R}\boldsymbol{y}\tilde{\boldsymbol{R}}$. We compute

$$\begin{aligned}(\boldsymbol{x}' \wedge \boldsymbol{y}') \cdot \boldsymbol{J} &= \langle \boldsymbol{x}' \wedge \boldsymbol{y}' \boldsymbol{J} \rangle = \langle \boldsymbol{R}\boldsymbol{x}\tilde{\boldsymbol{R}}\boldsymbol{R}\boldsymbol{y}\tilde{\boldsymbol{R}}\boldsymbol{J} \rangle = \langle \boldsymbol{x}\boldsymbol{y}\tilde{\boldsymbol{R}}\boldsymbol{J}\boldsymbol{R} \rangle, \\ &= (\boldsymbol{x} \wedge \boldsymbol{y}) \cdot (\tilde{\boldsymbol{R}}\boldsymbol{J}\boldsymbol{R}). \end{aligned} \tag{3.33}$$

This equation holds for all $\boldsymbol{x}$, $\boldsymbol{y}$ of 2n-D space, and thus the following equality has to be true:

$$\boldsymbol{J} = \tilde{\boldsymbol{R}}\boldsymbol{J}\boldsymbol{R}. \tag{3.34}$$

Since we are dealing with the rotor group which leaves the $\boldsymbol{J}$ invariant, this equality is satisfied. Unitary groups are constructed from reflections and rotations which leave $\boldsymbol{J}$ invariant. A reflection which satisfies the invariance constraint would require that a vector generator $\boldsymbol{u}$ fulfills

$$\boldsymbol{u}\boldsymbol{J}\boldsymbol{u} = \boldsymbol{J}. \tag{3.35}$$

This result implies that $\boldsymbol{u} \cdot \boldsymbol{J} = 0$; therefore, $(\boldsymbol{u} \cdot \boldsymbol{J}) \cdot \boldsymbol{J} = -\boldsymbol{u} = 0$. Thus, there are no vector generators of reflections, and we conclude that all unitary transformations are generated by elements of the spin group. Note that we have not specified so far the underlying signature, so these computing can be applied in the same way to the unitary groups $U(n)$ and $U(p, q)$ which in turn can be represented by using even multivectors of $G_{2n,0}$ and $G_{2p,2q}$, respectively.

A rotor is given by

$$\boldsymbol{R} = e^{-\frac{\boldsymbol{B}}{2}}, \tag{3.36}$$

where the bivector generators of the unitary group satisfy

$$\boldsymbol{B} \times \boldsymbol{J} = 0. \tag{3.37}$$

This defines the bivector realization of the Lie algebra of the unitary group, and it is called $u(n)$. In order to construct bivectors satisfying this relation, we resort to the Jacobi identity of Eq. (3.11) to prove that, c.f. [3, 11.110–111],

$$[(\boldsymbol{x} \cdot \boldsymbol{J}) \wedge (\boldsymbol{y} \cdot \boldsymbol{J})] \times \boldsymbol{J} = -(\boldsymbol{x} \cdot \boldsymbol{J}) \wedge \boldsymbol{y} + (\boldsymbol{y} \cdot \boldsymbol{J}) \wedge \boldsymbol{x} = -(\boldsymbol{x} \wedge \boldsymbol{y}) \times \boldsymbol{J}. \tag{3.38}$$

This leads to

$$[\boldsymbol{x} \wedge \boldsymbol{y} + (\boldsymbol{x} \cdot \boldsymbol{J}) \wedge (\boldsymbol{y} \cdot \boldsymbol{J})] \times \boldsymbol{J} = 0. \tag{3.39}$$

Note that a bivector similar to the form on the left-hand side will commute with J. Now, if we try all the combinations of the pair $\{e_i, f_i\}$, we find the following Lie algebra basis for $u(n)$:

$$\begin{aligned} \boldsymbol{J}_i &= e_i f_i && (i = 1, \ldots, n), \\ \boldsymbol{E}_{ij} &= e_i e_j + f_i f_j && (i < j = 1, \ldots, n), \\ \boldsymbol{F}_{ij} &= e_i f_j - f_i e_j && (i < j = 1, \ldots, n). \end{aligned} \tag{3.40}$$

All the bivectors belong to the geometric algebra $G_{2n,0}$, and the vectors $\{e_i\}$ and $\{f_i\}$ form an orthonormal basis for this geometric algebra. It is easy to establish the closure of this algebra under the commutator product. The doubling bivector $\boldsymbol{J}$ belongs to this algebra, and it commutes with all other elements of this algebra. The complex structure is generated by the bivector $\boldsymbol{J} = \boldsymbol{J}_1 + \boldsymbol{J}_2 + \cdots + \boldsymbol{J}_n$. Without considering the term $\boldsymbol{J}$, this algebra defines the special unitary group $SU(n)$. Similar analysis can be carried out for the base space of different signatures in order to ultimately build a bivector representation of the Lie algebra $u(p, q)$. These groups can be formulated in terms of even multivectors belonging to $G_{2n,0}$ or $G_{2p,2q}$, respectively.

3.5 Geometric Algebra of Reciprocal Null Cones

This section introduces the 2^{2n}-dimensional geometric algebra $G_{n,n}$. This geometric algebra is best understood by considering the properties of two 2^n-dimensional *reciprocal Grassmann subalgebras*. These subalgebras are generated by the vector bases of the reciprocal null cones N and $\overline{N}$. Let us start by explaining the meaning of null cones.

3.5.1 Reciprocal Null Cones

The *reciprocal null cones* N and $\overline{N}$ are real linear n-dimensional vector spaces whose vectors square to null—that is, for $\boldsymbol{x} \in N$, $\boldsymbol{x}^2 = 0$ and for $\overline{\boldsymbol{x}} \in \overline{N}$, $\overline{\boldsymbol{x}}^2 = 0$. In that regard, an associative geometric multiplication also equals zero—namely, for $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z} \in N$,

$$\boldsymbol{z}^2 = (\boldsymbol{x} + \boldsymbol{y})^2 = (\boldsymbol{x} + \boldsymbol{y})(\boldsymbol{x} + \boldsymbol{y}) = \boldsymbol{x}^2 + \boldsymbol{x}\boldsymbol{y} + \boldsymbol{y}\boldsymbol{x} + \boldsymbol{y}^2 = \boldsymbol{x}\boldsymbol{y} + \boldsymbol{y}\boldsymbol{x} = 0. \tag{3.41}$$

This result indicates that the symmetric part of geometric product of two vectors is simply zero. Thus, the geometric product of two vectors will be equal to the outer product:

$$\boldsymbol{x}\boldsymbol{y} = \boldsymbol{x} \cdot \boldsymbol{y} + \boldsymbol{x} \wedge \boldsymbol{y} = \frac{1}{2}(\boldsymbol{x}\boldsymbol{y} + \boldsymbol{y}\boldsymbol{x}) + \frac{1}{2}(\boldsymbol{x}\boldsymbol{y} - \boldsymbol{y}\boldsymbol{x}) = \frac{1}{2}(\boldsymbol{x}\boldsymbol{y} - \boldsymbol{y}\boldsymbol{x}) = \boldsymbol{x} \wedge \boldsymbol{y}. \tag{3.42}$$

Similarly, for the vectors in the reciprocal null cone,

$$\overline{\boldsymbol{x}\,\boldsymbol{y}} = \overline{\boldsymbol{x}} \cdot \overline{\boldsymbol{y}} + \overline{\boldsymbol{x}} \wedge \overline{\boldsymbol{y}} = \frac{1}{2}(\overline{\boldsymbol{x}\,\boldsymbol{y}} + \overline{\boldsymbol{y}\boldsymbol{x}}) + \frac{1}{2}(\overline{\boldsymbol{x}\,\boldsymbol{y}} - \overline{\boldsymbol{y}\boldsymbol{x}}) = \frac{1}{2}(\overline{\boldsymbol{x}\,\boldsymbol{y}} - \overline{\boldsymbol{y}\boldsymbol{x}}) = \overline{\boldsymbol{x}} \wedge \overline{\boldsymbol{y}}. \tag{3.43}$$

The *reciprocal vector bases* $\{w\}$ and $\{\overline{w}\}$ span the reciprocal null cones:

$$N = span\{w_1, \ldots, w_n\} \qquad \overline{N} = span\{\overline{w}_1, \ldots, \overline{w}_n\}. \tag{3.44}$$

The null spaces N or to the reciprocal and $\overline{N}$ have the following pseudoscalars:

$$W_n = w_1 w_2 \ldots w_n \qquad \bar{W}_n = \bar{w}_1 \bar{w}_2 \ldots \bar{w}_n. \tag{3.45}$$

The *neutral pseudo-Euclidean space* $\mathbb{R}^{n,n}$ is the linear space spanned by the null cones:

$$\mathbb{R}^{n,n} = span\{N, \overline{N}\} = \{\boldsymbol{x} + \overline{\boldsymbol{x}} | \boldsymbol{x} \in N, \overline{\boldsymbol{x}} \in \overline{N}\}. \tag{3.46}$$

The basis $\{w_i, \overline{w}_i\}$ is called a *Witt* basisin the theory of the quadratic forms. The reciprocal vector bases satisfy the following defining *inner product* relations:

$$w_i \cdot \overline{w}_j = \overline{w}_j \cdot w_i = \delta_{ij}, \tag{3.47}$$

and according to the symmetric relation of the vector inner product, it follows

$$w_i \overline{w}_j + \overline{w}_j w_i = \delta_{ij}, \tag{3.48}$$

also

$$w_i \cdot w_j = 0 \qquad \overline{w}_i \cdot \overline{w}_j = 0, \tag{3.49}$$

for all $i,\ j = 1, 2, \ldots, n$. The inner product relations of Eqs. (3.47) and (3.49) tell us that the reciprocal basis vectors w_i and $\overline{w}_i$ are all null vectors and are mutually orthogonal. Because of Eq. (3.47), the reciprocal vector bases are said to be dual, because they satisfy the relationship $\{w\} \cdot \{\overline{w}\} = id$, where id stands for the identity.

The outer product defining relations are

$$w_i \wedge w_j = w_i w_j = -w_j w_i \qquad \overline{w}_i \wedge \overline{w}_j = \overline{w}_i \overline{w}_j = -\overline{w}_j \overline{w}_i. \tag{3.50}$$

Note that the geometric product of two vectors equals the outer product only when both belong either to the null cone N or to the reciprocal cone $\overline{N}$. This is no longer true for arbitrary $\boldsymbol{x},\ \boldsymbol{y} \in \mathbb{R}^{n,n}$ owing to the dual relationship of the reciprocal bases expressed by Eqs. (3.49) and (3.50).

3.5.2 The Universal Geometric Algebra $G_{n,n}$

The vector bases of the reciprocal null cones N and $\overline{N}$ generate the 2^n-dimensional subalgebra $G_N = gen\{w_1, \ldots, w_n\}$ and the 2^n-dimensional reciprocal subalgebra $G_{\overline{N}} = gen\{\overline{w}_1, \ldots, \overline{w}_n\}$, which have the structure of Grassmann algebras.

The geometric algebra $G_{\bar{n},\bar{n}}$ is built by the direct product of these 2^n-dimensional Grassmann subalgebras:

$$G_{n,n} = G_N \otimes G_{\overline{N}} = gen\{w_1, \ldots, w_n, \overline{w}_1, \ldots, \overline{w}_n\}. \tag{3.51}$$

When n is countably infinite, we call $G_{\infty,\infty}$ the *universal geometric algebra*. The universal algebra $G_{\infty,\infty}$ contains all of the algebras $G_{n,n}$ as subalgebras.

The reciprocal bases $\{w\} \in N$ and $\{\bar{w}\} \in \bar{N}$ are also called $dual$, because they fulfill Eq. (3.47)—$\{w\} \cdot \{\bar{w}\} = id$. They generate the k-vector bases of $G_{\bar{n},\bar{n}}$,

$$\left\{1, \{w\}, \{\bar{w}\}, \{w_i w_j\}, \{\bar{w}_i \bar{w}_j\}, \{w_i \bar{w}_j\}, \ldots, \{w_{j_1} \ldots \bar{w}_{j_i} \bar{w}_{j_l} \ldots w_{j_k}\}, I, \bar{I}\right\}, \tag{3.52}$$

consisting of scalars, vectors, bivectors, trivectors, …, and the dual pseudoscalars $I = w_1 \wedge w_2 \wedge w_3 \ldots \wedge w_n$ and $\bar{I} = \bar{w}_1 \wedge \bar{w}_2 \wedge \bar{w}_3 \ldots \wedge \bar{w}_n$ which satisfy $I\bar{I} = 1$. Note that the $\begin{bmatrix} 2n \\ k \end{bmatrix}$-dimensional bases of k-vectors $\{w_{j_1} \ldots \bar{w}_{j_i} \bar{w}_{j_l} \ldots w_{j_k}\}$ for the $\begin{bmatrix} 2n \\ k \end{bmatrix}$ sets of indices $1 \leq j_1 < j_2 < \cdots < j_k \leq 2n$ are generated by different combinations of w's and $\bar{w}$'s.

3.5.3 Representations and Operations Using Bivector Matrices

In this subsection, we use a notation which extends the familiar addition and multiplication of matrices of real numbers to matrices consisting of vectors and bivectors. The notation is somewhat similar to the Einstein summation convention of tensor calculus, and it can be used to directly express the close relationships that exist between Clifford algebra and matrix algebra [4].

We begin by writing the *Witt basis* of null vector $\{w\}$ and the corresponding reciprocal basis $\{\bar{w}\}$ of $G_{\bar{n},\bar{n}}$ in *row form* and *column form*, respectively:

$$\{w\} = [w_1 w_2 \cdots w_n], \qquad \{\bar{w}\} = \begin{bmatrix} \bar{w}_1 \\ \bar{w}_2 \\ \vdots \\ \bar{w}_n \end{bmatrix}. \tag{3.53}$$

Taking advantage of the usual matrix multiplication between a *row* and a *column* and the properties of the geometric product, we get

$$\{\bar{w}\}\{w\} = \{\bar{w}\}\cdot\{w\} + \{\bar{w}\}\wedge\{w\} = \mathrm{I} + \begin{bmatrix} \bar{w}_1\wedge w_1 & \bar{w}_1\wedge w_2 & \dots & \bar{w}_1\wedge w_n \\ \bar{w}_2\wedge w_1 & \bar{w}_2\wedge w_2 & \dots & \bar{w}_2\wedge w_n \\ \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots \\ \bar{w}_n\wedge w_1 & \bar{w}_n\wedge w_2 & \dots & \bar{w}_n\wedge w_n \end{bmatrix}, \tag{3.54}$$

where I is the $n\times n$ identity matrix. Similarly,

$$\{w\}\{\bar{w}\} = \{w\}\cdot\{\bar{w}\} + \{w\}\wedge\{\bar{w}\} = \sum_{i=1}^{n} w_i\cdot\bar{w}_i + \sum_{i=1}^{n} w_i\wedge\bar{w}_i, \tag{3.55}$$

$$= n + \sum_{i=1}^{n} w_i\wedge\bar{w}_i. \tag{3.56}$$

In terms of the null cone bases, a vector $\boldsymbol{x}\in N$ is given by

$$\boldsymbol{x} = \{w\}\boldsymbol{x}_{\{w\}} = (w_1, w_2, \dots, w_n)\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \tag{3.57}$$

$$= (w_1, w_2, \dots, w_n)\begin{bmatrix} \overline{w}_1\cdot\boldsymbol{x} \\ \overline{w}_2\cdot\boldsymbol{x} \\ \vdots \\ \overline{w}_n\cdot\boldsymbol{x} \end{bmatrix} = \{w\}(\{\overline{w}\}\cdot\boldsymbol{x}) = \sum_{i=1}^{n} x_i w_i. \tag{3.58}$$

The vectors $\boldsymbol{x}\in N$ behave like column vectors, and the vectors $\overline{\boldsymbol{x}}\in N^*$ like row vectors. This property makes it possible to define the transpose of the vector $\boldsymbol{x}$ as follows:

$$\boldsymbol{x}^T = (\{w\}x_{\{w\}})^T = x_w^T\{\overline{w}\} = (x_1 x_2 \dots x_n)\begin{bmatrix} \overline{w}_1 \\ \overline{w}_2 \\ \vdots \\ \overline{w}_n \end{bmatrix}. \tag{3.59}$$

Note that using the transpose operation it is possible to move between the null cones N and N^*.

3.5.4 Bivector Representation of Linear Operators

One important application of the bivector matrix representation is the representation of a linear operator $f \in End(\mathcal{N})$. Recalling the basic Clifford algebra identity

$$(\boldsymbol{a} \wedge \boldsymbol{b}) \cdot \boldsymbol{x} = (\boldsymbol{b} \cdot \boldsymbol{x})\boldsymbol{a} - (\boldsymbol{a} \cdot \boldsymbol{x})\boldsymbol{b}, \tag{3.60}$$

between the bivector $\boldsymbol{a} \wedge \boldsymbol{b}$ and the vector $\boldsymbol{x}$, we can use the reciprocal bases to express the vector $\boldsymbol{x}$ in the form

$$\boldsymbol{x} = \{w\}\boldsymbol{x}_{\{w\}} = (\{w\} \wedge \{\bar{w}\}) \cdot \boldsymbol{x}, \tag{3.61}$$

where $\boldsymbol{x}_{\{w\}}$ represents the *column vector* components of $\boldsymbol{x}$:

$$\boldsymbol{x}_{\{w\}} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} \bar{w}_1 \cdot \boldsymbol{x} \\ \bar{w}_2 \cdot \boldsymbol{x} \\ \vdots \\ \bar{w}_n \cdot \boldsymbol{x} \end{bmatrix} = \{\bar{w}\} \cdot \boldsymbol{x}. \tag{3.62}$$

This is the *key idea* to the bivector representation of a linear operator. Let $f \in End(\mathcal{N})$, and we then have the following relationships:

$$\begin{aligned} f(\boldsymbol{x}) &= f(\{w\}\boldsymbol{x}_{\{w\}}) = \{w\}\mathcal{F}\boldsymbol{x}_{\{w\}} = \Big((\{w\}\mathcal{F}) \wedge \{\bar{w}\}\Big) \cdot \boldsymbol{x}, \\ &= F \cdot \boldsymbol{x}, \end{aligned} \tag{3.63}$$

where the bivector $F \in G$ is defined by

$$F = (\{w\}\mathcal{F}) \wedge \{\bar{w}\} = \sum_{i=1}^{n}\sum_{j=1}^{n} f_{ij} w_i \wedge \bar{w}_j = \{w\} \wedge (\mathcal{F}\{\bar{w}\}). \tag{3.64}$$

Thus, a linear operator $f \in End(\mathcal{N})$ can now be pictured as a linear mapping $f : \mathcal{N} \to \mathcal{N}$ of the null cone $\mathcal{N}$ onto itself. Furthermore, it can be represented in the bivector form $f(\boldsymbol{x}) = F \cdot \boldsymbol{x}$, where $F = (\{w\}\mathcal{F}) \wedge \{\bar{w}\}$ is a bivector in the enveloping geometric algebra G. As an example, we can show for the linear operator T its representation as a bivector matrix:

$$T = \begin{bmatrix} t_{11} & t_{12} & t_{13} & t_{14} \\ t_{21} & t_{22} & t_{23} & t_{24} \\ t_{31} & t_{32} & t_{33} & t_{34} \\ t_{41} & t_{42} & t_{43} & t_{44} \end{bmatrix} \tag{3.65}$$

$$\rightarrow \begin{bmatrix} t_{11} w_1 \wedge \bar{w}_1 & t_{12} w_1 \wedge \bar{w}_2 & t_{13} w_1 \wedge \bar{w}_3 & t_{14} w_1 \wedge \bar{w}_4 \\ t_{21} w_2 \wedge \bar{w}_1 & t_{22} w_2 \wedge \bar{w}_2 & t_{23} w_2 \wedge \bar{w}_3 & t_{24} w_2 \wedge \bar{w}_4 \\ t_{31} w_3 \wedge \bar{w}_1 & t_{32} w_3 \wedge \bar{w}_2 & t_{33} w_3 \wedge \bar{w}_3 & t_{34} w_3 \wedge \bar{w}_4 \\ t_{41} w_4 \wedge \bar{w}_1 & t_{42} w_4 \wedge \bar{w}_2 & t_{43} w_4 \wedge \bar{w}_3 & t_{44} w_4 \wedge \bar{w}_4 \end{bmatrix}. \tag{3.66}$$

Now, by considering $f, g \in gl(\mathcal{N})$ in the bivector form $f(\boldsymbol{x}) = F \cdot \boldsymbol{x}$ and $g(\boldsymbol{x}) = G \cdot \boldsymbol{x}$, and by calculating the commutator $[f, g]$, we find

$$[f, g](\boldsymbol{x}) = F \cdot (G \cdot \boldsymbol{x}) - G \cdot (F \cdot \boldsymbol{x}) = (F \times G) \cdot \boldsymbol{x}, \tag{3.67}$$

where the *commutator product of bivectors* $F \times G$ is defined by $F \times G = \frac{1}{2}[FG - GF]$. Thus the Lie bracket of the linear operators f and g becomes the commutator product of their respective bivectors F and G.

3.5.5 The Standard Bases of $G_{n,n}$

In the previous subsections, we have discussed in some detail the *orthogonal null vector bases* $\{w_i, \bar{w}_i\}$ of the geometric algebra $G_{\bar{n},\bar{n}}$. Using these orthogonal null vector bases, we can construct new *orthonormal* bases $\{e, \bar{e}\}$ of $G_{n,n}$,

$$\begin{aligned} e_i &= (w_i + \bar{w}_i), \\ \bar{e}_i &= (w_i - \bar{w}_i), \end{aligned} \tag{3.68}$$

for $i = 1, 2, 3, \ldots, n$. According to the properties of Eqs. (3.47)–(3.50), these basis vectors for $i \neq j$ satisfy

$$\begin{aligned} {e_i}^2 = 1,\ {\bar{e}_i}^2 = -1, \\ e_i \cdot e_j = \delta_{i,j},\ \bar{e}_i \cdot \bar{e}_j = -\delta_{i,j}, \\ e_i e_j = -e_j e_i,\ \bar{e}_i \bar{e}_j = -\bar{e}_j \bar{e}_i, \\ e_i \cdot \bar{e}_j = 0, \\ e_i \bar{e}_j = -e_j \cdot \bar{e}_i. \end{aligned} \tag{3.69}$$

The orthonormal vector basis $\{e_i\}$ spans a real Euclidean vector space $\mathbb{R}^n$, while the reciprocal orthonormal basis $\{\bar{e}_i\}$ spans the *anti*-Euclidean space $\overline{\mathbb{R}}^n$; hence, $\mathbb{R}^{n,n}$ can be expressed as the direct product of the Euclidean and anti-Euclidean spaces:

$$\mathbb{R}^{n,n} = \mathbb{R}^n \otimes \overline{\mathbb{R}}^n. \tag{3.70}$$

The basis $\{e_i\}$ generates the geometric subalgebra $G_{n,0}$, whereas $\{\bar{e}_i\}$ generates the geometric subalgebra $G_{0,n}$; thus, the geometric algebra $G_{n,n}$ is generated in terms of the bases $\{e_i\}$ and $\{\bar{e}_i\}$

$$G_{n,n} = gen\{e_1, \ldots, e_n, \bar{e}_1, \ldots, \bar{e}_n\}, \tag{3.71}$$

where the generated k-vectors are

$$\left\{1, \{e\}, \{\bar{e}\}, \{e_i e_j\}, \{\bar{e}_i \bar{e}_j\}, \{e_i \bar{e}_j\}, \{e_i e_j \bar{e}_k\}, \ldots, \{e_{j_1} \ldots \bar{e}_{j_i} \bar{e}_{j_l} \ldots e_{j_k}\}, I, \bar{I}\right\}, \tag{3.72}$$

consisting of scalars, vectors, bivectors, trivectors, …, and the dual pseudoscalars $I = e_1 \wedge e_2 \wedge e_3 \ldots \wedge e_n$ and $\bar{I} = \bar{e}_1 \wedge \bar{e}_2 \wedge \bar{e}_3 \ldots \wedge \bar{e}_n$. Note that the $\begin{bmatrix} 2n \\ k \end{bmatrix}$-dimensional bases of k-vectors $\{e_{j_1} \ldots \bar{e}_{j_i} \bar{e}_{j_l} \ldots e_{j_k}\}$ for the $\begin{bmatrix} 2n \\ k \end{bmatrix}$ sets of indices $1 \leq j_1 < j_2 < \cdots < j_k \leq 2n$ are generated by different combinations of e's and $\bar{e}$'s.

Using the basis $\{e_i, \bar{e}_i\}$, one can construct $(p+q)$-blades

$$E_{p,q} = E_p \bar{E}^\dagger = E_p \wedge \bar{E}^\dagger, \tag{3.73}$$

where $E_p = e_1 e_2 \ldots e_p = E_{p,0}$, $\bar{E}_p = \bar{e}_1 \bar{e}_2 \ldots \bar{e}_p = E_{0,p}$ and $(\cdot)^\dagger$ stands for the reversion operation. Note the each blade determines a projection $\underline{E} \in \mathbb{R}^{n,n}$ into a $(p+q)$-dimensional subspace $\mathbb{R}^{p,q}$ defined by, c.f. [2, 3.20–24],

$$\underline{E}_{p,q}(\boldsymbol{x}) = (\boldsymbol{x} \cdot E_{p,q}) E_{p,q}^{-1} = \frac{1}{2}[\boldsymbol{x} - (-1)^{p+q} E_{p,q} E_{p,q} \boldsymbol{x} E_{p,q}^{-1}]. \tag{3.74}$$

We use an underbar to distinguish linear operators from elements of the geometric algebra; in order to designate the operator by a multivector as in Eq. 3.74, the operator $\underline{E}_{p,q}$ is determined by the blade $E_{p,q}$.

The vector $\boldsymbol{x}$ is in the space $\mathbb{R}^{n,n}$ if and only if the wedge product is null

$$\boldsymbol{x} \wedge E_{p,q} = \boldsymbol{x} E_{p,q} + (-1)^{p+q} E_{p,q} \boldsymbol{x} = 0. \tag{3.75}$$

For the case of $q = 0$, Eq. (3.74) reads

$$\underline{E}_n(\boldsymbol{x}) = \frac{1}{2}(\boldsymbol{x} + \boldsymbol{x}^*), \tag{3.76}$$

and the involution of $\boldsymbol{x}$

$$\boldsymbol{x}^* = (-1)^{(n+1)} E_n \boldsymbol{x} E_n^{-1}. \tag{3.77}$$

3.5.6 Endomorphisms of $G_{n,n}$

This subsection is based on c.f. [2, Sect. V]. The universal algebra is a proper framework to handle the theory of linear transformations and Lie groups. This arises naturally as the endomorphism algebra $\mathcal{E}nd(G_n)$ which is the algebra of linear maps of the Euclidean geometric algebra G_n onto itself. This algebra is isomorphic to the algebra of real $2^n \times 2^n$ matrices, i.e.,

$$\mathcal{E}nd(G_n) \simeq M(2n, \mathbb{R}). \tag{3.78}$$

For an arbitrary multivector $\boldsymbol{M} \in G_n$, left and right multiplications by the basis vectors e_i determine endomorphisms of G_n defined as follows:

$$\begin{aligned} \underline{e}_i &: \boldsymbol{M} \to e_i(\boldsymbol{M}) = \underline{e}_i \boldsymbol{M}, \\ \bar{\underline{e}}_i &: \boldsymbol{M} \to \bar{\underline{e}}_i(\boldsymbol{M}) = \overline{\boldsymbol{M}} e_i, \end{aligned} \tag{3.79}$$

where the overbar stands for main involution of G_n,

$$\begin{aligned} \bar{e}_i &= -e_i, \\ \overline{\boldsymbol{M}_1 \boldsymbol{M}_2} &= \overline{\boldsymbol{M}}_1 \overline{\boldsymbol{M}}_2. \end{aligned} \tag{3.80}$$

The operators $\underline{e}_i$ are linear and they satisfy the Following:

$$\begin{aligned} \underline{e}_i \underline{e}_j + \underline{e}_j \underline{e}_i &= 2\delta_{ij}, \\ \bar{\underline{e}}_i \bar{\underline{e}}_j + \bar{\underline{e}}_j \bar{\underline{e}}_i &= -2\delta_{ij}, \\ \underline{e}_i \bar{\underline{e}}_j + \bar{\underline{e}}_j \underline{e}_i &= 0. \end{aligned} \tag{3.81}$$

Considering the inner product definition of G_n, these relations are isomorphic to the relations (3.69) for the vector basis $\{e_i, \bar{e}_i\} \in G_n$; this establishes the algebra isomorphism

$$G_{n,n} \simeq \mathcal{E}nd(G_n). \tag{3.82}$$

The equations to define this isomorphism were first formulated by Sommen and Van Acker [5] and included in [2].

Next, let us show that the universal algebra contains a $2n$-dimensional subspace $\mathcal{S}_n$ on which all the endomorphisms G_n are faithfully represented by left multiplication with elements belonging to $G_{n,n}$. The space $\mathcal{S}_n$ is known as *spinor space*, and its elements are called *spinors*. $\mathcal{S}_n$ is a *minimal left ideal* of $G_{n,n}$. Its construction can be described after establishing the necessary algebraic relations.

The spinor space is generated by multiplicating by the pseudoscalar I from the left by the entire mother algebra, as follows c.f. [2, 5.21]:

$$\mathcal{S}_n = G_{n,n} I. \tag{3.83}$$

As defined by Eq. (3.73), $E_{n,n} = E_n \bar{E}_n^{\dagger}$ is the unit pseudoscalar for $G_{n,n}$. Let us study linear transformations on $G_{n,n}$ and on its subspaces, particularly orthogonal transformations. The orthogonal transformation $\underline{R}$ is defined if it leaves the inner product invariant, namely,

$$(\underline{R}\boldsymbol{x}) \cdot (\underline{R}\boldsymbol{y}) = \boldsymbol{x} \cdot \boldsymbol{y}. \tag{3.84}$$

This linear transformation is called a $rotation$ if $det\ \underline{R} = 1$, i.e.,

$$\underline{R}(E_{n,n}) = E_{n,n}. \tag{3.85}$$

The rotations form a group known the *special orthogonal group* SO(n, n).

From the standard basis (3.68), the null basis vectors are

$$w_i = \frac{1}{2}(e_i + \bar{e}_i), \qquad \bar{w}_i = \frac{1}{2}(e_i - \bar{e}_i). \tag{3.86}$$

Using these null vectors, one constructs the commuting vectors I_i, which can be written as

$$I_i = \bar{w}_i w_i = (1 + e_i \bar{e}_i) = (1 + \boldsymbol{K}_i) = e_i w_i = \bar{w}_i e_i = w_i \bar{e}_i = -\bar{e}_i w_i, \tag{3.87}$$

for $i = 1, 2, \ldots, n$. The idempotent and commutativity properties of I_i and I are

$$\begin{aligned} &I_i^2 = I_i, \qquad I_i \times I_j = 0, \\ &I^2 = I. \end{aligned} \tag{3.88}$$

Multiplying I_i from the left or right by e_i, we get the next relations

$$\begin{aligned} e_i I_i &= \bar{e}_i I_i = w_i I_i = w_i = w_i \bar{w}_i w_i, \\ I_i e_i &= -I_i \bar{e}_i = I_i \bar{w}_i = \bar{w}_i = \bar{w}_i w_i \bar{w}_i. \end{aligned} \tag{3.89}$$

Using the I_i, one constructs the *universal idempotent*, c.f. [2, 5.14],

$$I = I_1 I_2 \ldots I_n = \overline{W}_n^{\dagger} W_n = \overline{W}_n W_n^{\dagger}, \tag{3.90}$$

where W_n is the pseudoscalar of the null space N.

Since $e_i I = \bar{e}_i I = w_i I$ and $E_{n,n} I = I$, where the unit pseudoscalar $E_{n,n} = E_n \bar{E}_n^{\dagger}$ for $G_{n,n}$, it follows, c.f. [2, 5.22],

$$E_{n,n} e_i I = -e_i I = E_{n,n} \bar{e}_i I = -\bar{e}_i I. \tag{3.91}$$

Thus, multiplication of $\mathcal{S}$ by $E_{n,n}$ corresponds to the main involution in G_n, written as

$$E_{n,n}\mathcal{S}_n \Longleftrightarrow \overline{G}_n. \tag{3.92}$$

Considering the relations of Eq. (3.81) and the definitions of (3.79), it follows

$$\begin{aligned} e_i\mathcal{S}_n &\longleftrightarrow e_i G_n, \\ \bar{e}_i\mathcal{S}_n &\longleftrightarrow \overline{G}_n e_i, \end{aligned} \tag{3.93}$$

where the over bar stands for the main involution, and using (3.91), one gets

$$\bar{e}_i E_{n,n}\mathcal{S}_n \longleftrightarrow G_n e_i. \tag{3.94}$$

One can easily prove that the reversion in $G_{n,n}$, corresponds to the reversion in G_n.

All these equations determine the interpretation of the spinor space operators as G_n endomorphisms.

3.5.7 The Bivectors of $G_{n,n}$

Let $\boldsymbol{K}$ any bivector in $\bigwedge^2(\mathbb{R}^{n,n})$ which can be expressed as sum of n commuting blades with unit square, c.f. [2, 3.25],

$$\boldsymbol{K} = \sum_{i=0}^{n} \boldsymbol{K}_i, \tag{3.95}$$

where

$$\boldsymbol{K}_i \times \boldsymbol{K}_j = 0, \tag{3.96}$$

$$\boldsymbol{K}_i^2 = 1. \tag{3.97}$$

The bivector $\boldsymbol{K}$ determines an automorphism of $\mathbb{R}^{n,n}$

$$\underline{K} : \boldsymbol{x} \to \bar{\boldsymbol{x}} = \underline{K}\boldsymbol{x} = \boldsymbol{x} \times \boldsymbol{K} = \boldsymbol{x} \cdot \boldsymbol{K}. \tag{3.98}$$

This is a mapping of a vector $\boldsymbol{x}$ into a vector $\bar{\boldsymbol{x}}$, which is called the complement of $\boldsymbol{x}$ with respect to $\boldsymbol{K}$. It is easy to prove that $\underline{K}\bar{\boldsymbol{x}} = \underline{K}^2\boldsymbol{x} = \boldsymbol{x}$ or as an operator equation $\underline{K}^2 = 1$; therefore, $\underline{K}$ is an involution. In addition,

$$\begin{aligned} \boldsymbol{x} \cdot \bar{\boldsymbol{x}} &= 0, \\ \boldsymbol{x}_\pm &= \boldsymbol{x} \pm \bar{\boldsymbol{x}} = \boldsymbol{x} \pm \boldsymbol{x} \cdot \boldsymbol{K}, \end{aligned} \tag{3.99}$$

are null vectors. As a matter of fact, the sets $\{\boldsymbol{x}_+\}$ and $\{\boldsymbol{x}_-\}$ of all such vectors are dual n-dimensional vector spaces, and thus $\boldsymbol{K}$ determines the desired null space decomposition of the form (3.70) without referring to a vector basis.

A suitable bivector $\boldsymbol{K}$ can be computed using for the $\boldsymbol{K}_i$ the basis $\{e_i, \bar{e}_i\}$

$$\boldsymbol{K}_i = e_i \bar{e}_i = e_i \wedge \bar{e}_i. \tag{3.100}$$

This bivector $\boldsymbol{K}$ has the following properties:

$$\begin{aligned} e_i \times \boldsymbol{K} &= e_i \cdot \boldsymbol{K} = e_i(e_j \wedge \bar{e}_j) = e_i \cdot (e_j \wedge \bar{e}_j) = \bar{e}_i, \\ \bar{e}_i \times \boldsymbol{K} &= \bar{e}_i \cdot \boldsymbol{K} = \bar{e}_i(e_j \wedge \bar{e}_j) = -\bar{e}_i \cdot \bar{e}_j e_i = e_i, \end{aligned} \tag{3.101}$$

thus

$$(\boldsymbol{x} \cdot \boldsymbol{K}) \cdot \boldsymbol{K} = \boldsymbol{K} \cdot (\boldsymbol{K} \cdot \boldsymbol{x}) = \boldsymbol{x} \quad \forall \boldsymbol{x}. \tag{3.102}$$

The difference between the doubling bivector $\boldsymbol{J}$ of Sect. 3.4 and $\boldsymbol{K}$ is crucial, because $\boldsymbol{J}$ generates a complex structure and in contrast $\boldsymbol{K}$ generates instead a *null* structure. In order to see clearly this, let us take any vector $\boldsymbol{x} \in G_{n,n}$ and define

$$\boldsymbol{x}_\pm = \boldsymbol{x} \pm \boldsymbol{x} \cdot \boldsymbol{K}, \tag{3.103}$$

taking its square

$$\begin{aligned} \boldsymbol{x}_\pm^2 &= \boldsymbol{x}^2 \pm 2\boldsymbol{x} \cdot (\boldsymbol{x} \cdot \boldsymbol{K}) + (\boldsymbol{x} \cdot \boldsymbol{K})^2 = \boldsymbol{x}^2 - \langle \boldsymbol{x} \cdot \boldsymbol{K}\boldsymbol{K}\boldsymbol{x} \rangle, \\ &= \boldsymbol{x}^2 - [(\boldsymbol{x} \cdot \boldsymbol{K}) \cdot \boldsymbol{K}] \cdot \boldsymbol{x} = \boldsymbol{x}^2 - \boldsymbol{x}^2 = 0, \end{aligned} \tag{3.104}$$

as expected by a null vector. Recalling the concept of projective split of Sect. 2.3.3, in a bit similar fashion here the bivector $\boldsymbol{K}$ splits vectors in $\boldsymbol{x} \in G^1_{n,n}$ into two separate null vectors, c.f. [3, 11.118],

$$\boldsymbol{x} = \boldsymbol{x}_+ + \boldsymbol{x}_-, \tag{3.105}$$

where

$$\boldsymbol{x}_+ = \frac{1}{2}(\boldsymbol{x} + \boldsymbol{x} \cdot \boldsymbol{K}), \qquad \boldsymbol{x}_- = \frac{1}{2}(\boldsymbol{x} - \boldsymbol{x} \cdot \boldsymbol{K}). \tag{3.106}$$

We see that the space of vectors $G^1_{n,n}$ decomposes into a direct sum of two null spaces: N, N^*, where $\boldsymbol{x}_+ \in N$. The vectors in N fulfill, c.f. [3, 11.119],

$$\boldsymbol{x}_+ \cdot \boldsymbol{K} = \boldsymbol{x}_+ \qquad \forall \boldsymbol{x}_+ \in N. \tag{3.107}$$

According to Eq. (3.104), we can see that all vectors $\boldsymbol{x}_+ \in N$ also square to zero. The space N defines a *Grassmann algebra*.

3.5.8 Mappings Acting on the Null Space

Every linear functions $\boldsymbol{x} \to \mathtt{f}(\boldsymbol{x})$ acting on an n-dimensional vector space can be represented in the null space N using an operator $\boldsymbol{O}$ as follows:

$$x_+ \to \boldsymbol{O}\boldsymbol{x}_+\boldsymbol{O}^{-1}, \tag{3.108}$$

where $\boldsymbol{O}$ is built by the geometric product on an *even* number of unit vectors. Here, vectors $\boldsymbol{x} \in G_n$ are mapped to null vectors $\boldsymbol{x}_+$ in $G_{n,n}$ which in turn are acted on by the multivector $\boldsymbol{O}$ so that the following is true:

$$\mathtt{f}(\boldsymbol{x}) + \mathtt{f}(\boldsymbol{x}) \cdot \boldsymbol{K} = \boldsymbol{O}(\boldsymbol{x} + \boldsymbol{x} \cdot \boldsymbol{K})\boldsymbol{O}^{-1}. \tag{3.109}$$

This equation defines a map between linear functions $\mathtt{f}(\boldsymbol{x})$ and multivectors $\boldsymbol{O} \in G_{n,n}$. Note that the map is not quite an isomorphism, due to the fact that both $\boldsymbol{O}$ and $-\boldsymbol{O}$ generate the same function, i.e., $\boldsymbol{O}$ forms a *double-cover* representation. The map will work, only if the action of $\boldsymbol{O}$ does not leave the space N, for that $\boldsymbol{O}$ must fulfill Eq. (3.107)

$$(\boldsymbol{O}\boldsymbol{x}_+\boldsymbol{O}^{-1}) \cdot \boldsymbol{K} = \boldsymbol{O}\boldsymbol{x}_+\boldsymbol{O}^{-1}. \tag{3.110}$$

After simple algebraic manipulation involving inner product between vectors and bivectors, c.f. [3, 11.123–124],

$$\begin{aligned} \boldsymbol{x}_+ &= \boldsymbol{O}^{-1}(\boldsymbol{O}\boldsymbol{x}_+\boldsymbol{O}^{-1}) \cdot \boldsymbol{K}\boldsymbol{O}, \\ &= \boldsymbol{O}^{-1}\frac{1}{2}(\boldsymbol{O}\boldsymbol{x}_+\boldsymbol{O}^{-1}\boldsymbol{K} - \boldsymbol{K}\boldsymbol{O}\boldsymbol{x}_+\boldsymbol{O}^{-1})\boldsymbol{O}, \\ &= \boldsymbol{x}_+ \cdot (\boldsymbol{O}^{-1}\boldsymbol{K}\boldsymbol{O}). \end{aligned} \tag{3.111}$$

In this equation, according to Eq. (3.107), it follows that it is required $\boldsymbol{O}^{-1}\boldsymbol{K}\boldsymbol{O} = \boldsymbol{K}$ or

$$\boldsymbol{O}\boldsymbol{K} = \boldsymbol{K}\boldsymbol{O}. \tag{3.112}$$

Since $\boldsymbol{O}$ is built by a product of even number of unit vectors, it follows $\boldsymbol{O}\boldsymbol{O}^{-1} = \pm 1$. The subgroup which fulfills $\boldsymbol{O}\boldsymbol{O}^{-1} = 1$ are rotors in $G_{n,n}$, and their generators (Lie algebra basis elements) are bivectors. Thus the condition $\boldsymbol{R}\boldsymbol{K}\tilde{\boldsymbol{R}} = \boldsymbol{K}$ is the direct analog of the condition that defined the unitary group expressed in terms of rotors which leaves in this case $\boldsymbol{J}$ invariant, see Eq. (3.34).

3.5.9 The Bivector or Lie Algebra of $G_{n,n}$

The bivector or Lie algebra generators are the set of bivectors that commute with $\boldsymbol{K}$. The Jacobi identity of Eq. (3.11) guarantees that the commutator of two bivectors which commute with $\boldsymbol{K}$ yields a third which in turn also commutes with $\boldsymbol{K}$. Similar as by the case of the unitary group of Eq. (3.11), [3, 11.125], formulate the following algebraic constraint:

$$[(\boldsymbol{x} \cdot \boldsymbol{K}) \wedge (\boldsymbol{y} \cdot \boldsymbol{K})] \times \boldsymbol{K} = \boldsymbol{x} \wedge (\boldsymbol{y} \cdot \boldsymbol{K}) + (\boldsymbol{x} \cdot \boldsymbol{K}) \wedge y = (\boldsymbol{x} \wedge \boldsymbol{y}) \times \boldsymbol{K}, \tag{3.113}$$

after passing the far most right element to the left of the equation

$$[\boldsymbol{x} \wedge \boldsymbol{y} - (\boldsymbol{x} \cdot \boldsymbol{K})(\boldsymbol{y} \cdot \boldsymbol{K})] \times \boldsymbol{K} = 0. \tag{3.114}$$

By using this constraint, we can again try all combinations of $\{e_i, \bar{e}_i\}$ to produce the bivector basis for the Lie algebra $gl(n)$ of the general linear group $GL(n)$, c.f. [2, 4.29a–c] and c.f. [3, Table 11.2],

$$\boldsymbol{K}_i = \frac{1}{2}\boldsymbol{F}_{ij} = e_i \bar{e}_i, \tag{3.115}$$

$$\boldsymbol{E}_{ij} = e_i e_j - \bar{e}_i \bar{e}_j \qquad (i < j = 1, \ldots, n), \tag{3.116}$$

$$\boldsymbol{F}_{ij} = e_i \bar{e}_j - \bar{e}_i e_j \qquad (i < j = 1, \ldots, n). \tag{3.117}$$

The $\{e_i\}$ basis vectors are orthonormal with positive signature, and the $\{\bar{e}_i\}$ are orthonormal with negative signature and belong to the geometric algebra $G_{n,n,}$. This algebra contains the bivector $\boldsymbol{K} = \boldsymbol{K}_1 + \boldsymbol{K}_2 + \cdots + \boldsymbol{K}_n$ which generates the Abelian subgroup of global dilation. By factoring out this bivector, one gets the special Lie algebra $sl(n)$. The special linear group $SL(n)$ can be characterized as the group of volume and orientation preserving linear transformations of $\mathbb{R}^n$.

Note that the difference in structure between the Lie algebra $gl(n)$ of the linear group $GL(n)$ and the unitary group $U(n)$ is only due to the different signatures of their underlying spaces. By the case of the conformal geometric algebras presented in Chap. 6, the Lie algebras are related to a space with pseudo-Euclidean signature.

3.6 The General Linear Group as Spin Group

This section is based on the work of Hestenes and Sobzyk [1, Chap. 8], a late article of Doran et al. [2, Sect. IV], and the Doran's Ph.D. thesis [6].

As shown in Sect. 3.3.1, every rotor can be expressed in an exponential form

$$\boldsymbol{R} = e^{-\frac{1}{2}\boldsymbol{B}}, \tag{3.118}$$

where the bivector $\boldsymbol{B}$ is called the generator of $\boldsymbol{R}$. Every bivector determines a unique rotation. The bivectors are generators of a spin or rotation Lie group $Spin(n, n)$. The Lie algebra of $SO(n, n)$ and $Spin(n, n)$ is designed by the lowercase notations $so(n, n)$. The whole set of bivectors constitute a bivector basis which spans the linear space $\bigwedge^2 \mathbb{R}^{n,n}$ and under the commutator forms a Lie algebra. Note that every Lie algebra is a subalgebra of $so(n, n)$.

Lie groups are classified according to their invariants. For case of the classical groups, the invariants are non-degenerate bilinear or quadratic forms. The geometric algebra framework offers us a much more simpler alternative of invariants formulated in terms of multivectors which shape the bilinear forms. According to [1], every bilinear form can be written as, c.f. [2, 4.19],

$$\boldsymbol{x} \cdot (\underline{Q}\boldsymbol{y}) = \boldsymbol{x} \cdot (\boldsymbol{y} \cdot \underline{Q}) = (\boldsymbol{x} \wedge \boldsymbol{y}) \cdot \boldsymbol{Q}, \tag{3.119}$$

where $\boldsymbol{Q}$ is a bivector, and $\underline{Q}$ is the corresponding linear operator and the form is non-degenerated if $\underline{Q}$ is nonsingular, i.e., $\det \underline{Q} \neq 0$ and

$$\underline{Q}^2 = \pm\underline{1}. \tag{3.120}$$

Due to the fact that $(\underline{R}\boldsymbol{x} \wedge \underline{R}\boldsymbol{y}) \cdot \boldsymbol{Q} = (\boldsymbol{x} \wedge \boldsymbol{y}) \cdot \underline{R}^\dagger \boldsymbol{Q}$ an invariance of Eq. (3.119), it is equivalent to the so-called *stability condition*, c.f. [2, 4.21],

$$\underline{R}^\dagger \wedge \boldsymbol{Q} = \boldsymbol{R}^\dagger \boldsymbol{Q} \boldsymbol{R} = \boldsymbol{Q}. \tag{3.121}$$

This can explained as that the invariance group of any skew-symmetric bilinear form is the *stability group* of a bivector. According to Eq. (3.121), the generators of the stability group $G(\boldsymbol{Q})$ for $\boldsymbol{Q}$ necessarily must commute with $\boldsymbol{Q}$. In this regard, let us study the commutator of $\boldsymbol{Q}$ with an arbitrary two-blade $\boldsymbol{x} \wedge \boldsymbol{y}$. The Jacobi identity, Eq. (3.11), in terms of bivectors of Eq. (3.11) can be rewritten using $\boldsymbol{x} \wedge \boldsymbol{y}$ as follows, c.f. [2, 4.22–25]:

$$(\boldsymbol{x} \wedge \boldsymbol{y}) \times \boldsymbol{Q} = (\boldsymbol{x} \times \boldsymbol{Q}) \wedge \boldsymbol{y} + \boldsymbol{x} \wedge (\boldsymbol{y} \times \boldsymbol{Q}) = (\underline{Q}\boldsymbol{x}) \wedge \boldsymbol{y} + \boldsymbol{x} \wedge (\underline{Q}\boldsymbol{y}), \tag{3.122}$$

which leads to

$$\begin{aligned} [(\boldsymbol{x} \wedge \boldsymbol{y}) \times \boldsymbol{Q}] \times \boldsymbol{Q} &= [(\underline{Q}\boldsymbol{x}) \wedge \boldsymbol{y} + \boldsymbol{x} \wedge (\underline{Q}\boldsymbol{y})] \times \boldsymbol{Q}, \\ &= (\underline{Q}^2\boldsymbol{x}) \wedge \boldsymbol{y} + 2\underline{Q}(\boldsymbol{x} \wedge \boldsymbol{y}) + \boldsymbol{x} \wedge (\underline{Q}^2\boldsymbol{y}), \end{aligned} \tag{3.123}$$

using the condition of Eq. (3.120); for any bivector $\boldsymbol{B}$ using linearity, we can extend Eq. (3.123) to

$$(\boldsymbol{B} \times \boldsymbol{Q}) \times \boldsymbol{Q} = 2(\underline{Q}\boldsymbol{B} \pm \boldsymbol{B}). \tag{3.124}$$

Henceforth, if $\boldsymbol{B}$ commutes with $\boldsymbol{Q}$, we obtain

$$\underline{Q}\boldsymbol{B} = \mp\boldsymbol{B}. \tag{3.125}$$

Note that the signs are opposite to those of Eq. (3.120). This result tells us that the generators of $G(\boldsymbol{Q})$ are simply eigenvectors of $\underline{Q}$ with eigenvalues ∓ 1. We understand invariance under a Lie group rotation expressed as linear operator $\underline{R}$, if the isometry is preserved, c.f. [2, 4.10–13], namely,

$$(\underline{R}\boldsymbol{x}) \cdot (\underline{Q}\underline{R}\boldsymbol{y}) = \boldsymbol{x} \cdot (\underline{Q}\boldsymbol{y}), \tag{3.126}$$

$$\boldsymbol{x} \cdot (\underline{R}^{\dagger})\underline{Q}\underline{R}\boldsymbol{y}) = \boldsymbol{x} \cdot (\underline{Q}\boldsymbol{y}). \tag{3.127}$$

As an operator equation, this condition reads

$$\underline{R}^{\dagger}\underline{Q}\underline{R} = \underline{Q} = \underline{R}\underline{Q}\underline{R}^{\dagger} \tag{3.128}$$

or equivalent

$$\underline{Q}\underline{R} = \underline{R}\underline{Q}. \tag{3.129}$$

So the invariance group of the quadratic form $\underline{Q}$ consists of those rotations which commute with Q. In order to illustrate, let us consider the bilinear form $\boldsymbol{x} \cdot \boldsymbol{y}^*$ determined by Eq. (3.77) for involution which distinguishes the subspaces $\mathbb{R}^n$ and $\overline{\mathbb{R}}^n$. By applying the rotation group on $E_n = e_1 e_2 \ldots e_n$, we get

$$\underline{R}E_n = \boldsymbol{R}E_n\tilde{\boldsymbol{R}} = E_n. \tag{3.130}$$

Thus, invariance of the bilinear form $\boldsymbol{x} \cdot \boldsymbol{y}^*$ is equivalent to invariance of the n-blade E_n. Consequently, one can construct a basis for the Lie algebra from the vector basis $\{e_i, \bar{e}_i\}$ for the involution operator $*$, namely,

$$\begin{aligned} e_{ij} &= e_i e_j, \qquad && \texttt{for } i < j = 1, 2, \ldots, n, \\ \bar{e}_{kl} &= \bar{e}_k \bar{e}_l, \qquad && \texttt{for } k < l = 1, 2, \ldots, n. \end{aligned} \tag{3.131}$$

Any generator $\boldsymbol{B}$ in the geometric algebra can be expressed as follows, c.f. [2, 4.10–16]:

$$\boldsymbol{B} = \sum_{i<j} \alpha^{ij} e_{ij} + \sum_{i<j} \beta^{ij} \bar{e}_{ij}. \tag{3.132}$$

These Lie algebra generators form the corresponding Lie group or a rotor, c.f. [2, 4.10–18],

$$\boldsymbol{R} = e^{\boldsymbol{B}} = e^{\frac{1}{2}\sum_{i<j} \alpha^{ij} e_{ij} + \sum_{i<j} \beta^{ij} \bar{e}_{ij}} = e^{\frac{1}{2}\sum_{i<j} \alpha^{ij} e_{ij}} e^{\frac{1}{2}\sum_{i<j} \beta^{ij} \bar{e}_{ij}}. \tag{3.133}$$

The $\boldsymbol{R}$ is the spin representation for the product group $\mathrm{SO}(n) \otimes \mathrm{SO}(n)$. Since it is determined by the invariance of E_n given in Eq. (3.126), $\boldsymbol{R}$ is called to be the stability group of E_n. Here, for its characterization, it is not needed a direct reference to a quadratic form.

As an illustration, we identify Q with the complementation bivector $\boldsymbol{K}$ of Eq. (3.95) and from Eq. (3.97) $\boldsymbol{K}^2 = 1$. We select an orthonormal basis which factors the component blades $\boldsymbol{K}_i$ into orthogonal factors as in Eq. (3.100). Now, using Eq. (3.101), one obtains straightforwardly the generator basis given by Eqs. (3.115)–(3.117) for the stability group of $\boldsymbol{K}$. The stability group of $\boldsymbol{K}$ can be identified as the general linear group $\mathrm{GL}(n, \mathbb{R})$, for this proof see [2].

Equations (3.115)–(3.117) can be then used to represent every positive, nonsingular linear transformation in a spinor form, c.f. [2, 4.34], namely,

$$\boldsymbol{R} = e^{B} = e^{\frac{1}{2}\sum_{i<j}\alpha^{ij}E_{ij}+\sum_{i<j}\beta^{ij}F_{ij}+\sum_{i<j}\gamma^{ij}K_{ij}}, \tag{3.134}$$

$$= e^{\frac{1}{2}\sum_{i<j}\alpha^{ij}E_{ij}}e^{\frac{1}{2}\sum_{i<j}\beta^{ij}F_{ij}}e^{\frac{1}{2}\sum_{i<j}\gamma^{ij}K_{ij}}. \tag{3.135}$$

Equation (3.135) is an efficient representation of a composition of linear transformations in terms of the product of specific spinors, each one to handle certain transformation. The computation of composite rotations with a *spin representation* is much more efficient than the use of standard matrix based methods, particularly due to the redundancy in the matrix coefficients. For example, a 3D rotation matrix has nine real coefficients, in contrast a rotor just 4. Furthermore, the spinor representation applied to geometric entities conveys a clear geometric interpretation of the operation. From other perspective, for rotations, spinors in exponential form are not optimal for computational purposes; however, they are very attractive for Lie algebra analysis. Since the Lie algebra is a linear space, one can use the exponent variables for efficient algorithms in control engineering and neurocomputing.

3.6.1 The Special Linear Group SL(n, $\mathbb{R}$)

Continuing with our analysis of the group theory, one notes that $\boldsymbol{K}$ commutes all other elements of the Lie algebra $gl(n, \mathbb{R})$, and thus it generates a one-dimensional invariant subgroup of $GL(n, \mathbb{R})$. We can remove $\boldsymbol{K}$ from the group by replacing the $\boldsymbol{K}_i$ in Eq. (3.115) by, c.f. [2, 4.51],

$$\boldsymbol{H}_i = \boldsymbol{K}_i - \boldsymbol{K}_{i+1} \qquad (i = 1, 2, \ldots, n-1), \tag{3.136}$$

$\boldsymbol{H}_{ij}$ together with $\boldsymbol{E}_{ij}$ Eq. (3.116) and $\boldsymbol{F}_{ij}$ Eq. (3.117), these bivectors generate the *special linear group* $\mathrm{SL}(n, \mathbb{R})$, the subgroup of $\mathrm{GL}(n, \mathbb{R})$ for which the determinant is unity.

Next, we compose the complementation operator $\underline{K}$ with the operator (3.77) to formulate an operator $\underline{K}_*$ defined by

$$\underline{K}_* \boldsymbol{x} = \underline{K} \boldsymbol{x}^* = (\boldsymbol{x}^*) \cdot \boldsymbol{K}. \tag{3.137}$$

Consequently

$$\underline{K}_* e_j = \bar{e}_j, \qquad \underline{K}_* \bar{e}_j = -e_j, \tag{3.138}$$

thus

$$\underline{K}_*^2 = -1. \tag{3.139}$$

In analogy with Eq. (3.125), $\underline{K}_*$ defines a new Lie algebra with generators determined by the outermorphism condition

$$\underline{K}_*(\boldsymbol{B}) = -\boldsymbol{B}. \tag{3.140}$$

Using this result, one can construct a generator basis for an invariance group of $\underline{K}_*$ as follows, c.f. [2, 4.56–57]:

$$\boldsymbol{E}_{ij} = e_i e_j - \bar{e}_i \bar{e}_j, \tag{3.141}$$
$$\boldsymbol{F}_{ij} = e_i \bar{e}_j + \bar{e}_i e_j, \tag{3.142}$$

for $i, j = 1, 2, \ldots, n, i < j$. These are the generators of the *complex orthogonal group* $\mathrm{SO}(n, \mathbb{C})$. According to [2] for odd n, $\boldsymbol{K}$ is the only kind of involutory bivector. For the case when n is even, their invariants determine other groups, see [2, Sect. VI].

The general linear group $\mathrm{GL}(p, q)$ can be formulated and study essentially in the same manner as the Euclidean case previously explained, i.e., E_n is replaced by $E(p, q)$ and the corresponding pseudoscalar I_n is replaced by $I_{p,q} = w_1 \ldots w_p \bar{w}_{p+1} \ldots \bar{w}_n$.

3.6.2 The Pin(n) and Spin(n) Groups

Is worth to study the endomorphism correspondence for GL(n, $\mathbb{R}$), we start with the orthogonal group. Considering any unit vectors $u, v, \ldots, \in G_n$, Eq. (3.93) yields the following correspondences:

$$\begin{aligned} u\bar{u} S_n &\longleftrightarrow u G_{0,n} u, \\ v\bar{v}u\bar{u} S_n &\longleftrightarrow vu G_{n,0} uv. \end{aligned} \tag{3.143}$$

Equation (3.143) is (the outermorphism of) a reflection in $G_{\bar{n}}$ and Eq. (3.143), a double reflection, is a rotation in the $u \wedge v$ plane. The generalization is straightforward; for k-vectors $u_1, u_2, \ldots, u_k \in G_{\bar{n}}$, it follows, c.f. [2, 5.29–32],

$$U_{(k)} = u_k u_{k-1} \cdots u_2 u_1 \quad \text{and} \quad \overline{U} = \bar{u}_k \bar{u}_{k-1} \cdots \bar{u}_2 \bar{u}_1, \tag{3.144}$$

thus

$$\begin{aligned} U_{(k)} \overline{U}_{(k)} &= u_k \cdots u_1 \bar{u}_k \cdots \bar{u}_1 = (-1)^{k-1} u_k \cdots u_2 \bar{u}_k \cdots \bar{u}_2 u_1 \bar{u}_1 \cdots \\ &= \epsilon u_k \bar{u}_k u_{k-1} \cdots u_2 \bar{u}_2 u_1 \bar{u}_1, \end{aligned} \tag{3.145}$$

where $\epsilon = (-1)^{\frac{1}{2}k(k-1)}$. For an odd k value, one gets

$$\epsilon_k U_{(k)} \overline{U}_{(k)} S_n \longleftrightarrow \overline{U}_{(k)} \overline{G}_n U^{\dagger}_{(k)}, \tag{3.146}$$

and for even k value,

$$\epsilon_k U_{(k)} \overline{U}_{(k)} S_n \longleftrightarrow U_{(k)} G_n U^{\dagger}_{(k)}. \tag{3.147}$$

Equations (3.146) and (3.147) describe the complete orthogonal group $\mathrm{O}(n)$ as an automorphism group of G_n.

The successive reflections using unit vectors as shown in Eq. (3.144) build a multiplicative groups in $G_{\bar{n}}$, exemplified by U_k of Eq. (3.144); this group is called the Pin group G_n and denoted by $\mathrm{Pin}(n)$. The $\mathrm{Pin}(n)$ is a double covering of $\mathrm{O}(n)$. For n even, the subgroup is called $\mathrm{Spin}(n)$, which is the double covering group of $\mathrm{SO}(n)$.

If we use the notation of Eq. (3.119) and considering a pseudoscalar even I, the $U_{(k)}$ in Eq. (3.144) which are continuously connected with the identity, can be rewritten in a exponential form as follows, c.f. [2, 5.33]:

$$U_{(k)} = e^{\frac{1}{2}\sum_{i<j} \alpha^{ij} e_{ij}}, \tag{3.148}$$

henceforth

$$\begin{aligned} \epsilon_k U_{(k)} \overline{U}_{(k)} &= e^{\frac{1}{2}\sum_{i<j} \alpha^{ij} e_{ij}} e^{-\frac{1}{2}\sum_{i<j} \alpha^{ij} \bar{e}_{ij}} = e^{\frac{1}{2}\sum_{i<j} \alpha^{ij} (e_{ij} - \bar{e}_{ij})}, \\ &= e^{\frac{1}{2}\sum_{i<j} \alpha^{ij} E_{ij}} = e^{E}. \end{aligned} \tag{3.149}$$

The bivector generators $E_{ij} = e_i e_j - \bar{e}_i \bar{e}_j$ represent the actual left and right multiplications in G_n as presented in Eq. (3.146). The exponent of the bivector $E = \frac{1}{2}\sum_{i<j} \alpha^{ij} E_{ij}$ determines the rotor representation of $\mathrm{O}(n)$ given previously by Eq. (3.141).

3.6.2.1 Corollary

Previous subsections and section V of [2] show that projections as well as orthogonal and symmetric transformations on $\mathbb{R}^n$ can be represented in $G_{n,n}$ as simply even monomials, i.e., products of an even number of vectors which in turn cause successive

reflections. Hence, we can draw a key conclusion: *every linear transformation* in $\mathbb{R}^n$ can be represented in $G_{n,n}$ as an even multivector which commutes with the complementation bivector $\boldsymbol{K}$. This is quite useful, because the composition of linear transformations can be simply formulated in terms of geometric products among idempotents and rotors in Spin(n, n). Many equation manipulations are simplified, thanks to the commutativity with $\boldsymbol{K}$, as it is implicit in the reordering of vectors as shown in Eq. (3.145).

3.7 The $Pin^{sp}(3,3)$

In this subsection, we will formulate the Lie group and Lie algebra of the 3D projective geometry using the geometric algebra framework; specifically, we will model the Lie group $SL(4)$ and its Lie algebra $sl(4)$ in $G_{3,3}$.

According to Sect. 3.5.2, the Witt-vector bases of the reciprocal null cones N_3 and $\overline{N}_3$ build $\mathbb{R}^{3,3} = N_3 \oplus \overline{N}_3$ and generate the 2^3-dimensional subalgebra $G_N = gen\{w_1, w_2, w_3\}$ and the 2^3-dimensional reciprocal subalgebra $G_{\overline{N}} = gen\{\overline{w}_1, \overline{w}_2, \overline{w}_3\}$, which have the structure of Grassmann algebras.

The geometric algebra $G_{3,3}$ is built by the direct product of these 2^3-dimensional Grassmann subalgebras:

$$G_{3,3} = G_N \otimes G_{\overline{N}} = gen\{w_1, w_2, w_3, \overline{w}_1, \overline{w}_2, \overline{w}_3\}. \tag{3.150}$$

The reciprocal bases $\{w\} \in N$ and $\{\bar{w}\} \in \bar{N}$ are called *dual*, because they fulfill Eq. (3.47) $w_i \cdot \overline{w}_j = \overline{w}_j \cdot w_i = \delta_{ij}$. They generate the k-vector bases of $G_{3,3}$,

$$\left\{1, \{w\}, \{\bar{w}\}, \{w_i w_j\}, \{\bar{w}_i \bar{w}_j\}, \{w_i \bar{w}_j\}, \ldots, \{w_{j_1} \ldots \bar{w}_{j_i} \bar{w}_{j_l} \ldots w_{j_k}\}, I, \bar{I}\right\} \tag{3.151}$$

consisting of scalars, vectors, bivectors, trivectors, …, pseudoscalar and the dual pseudoscalars $I = w_1 \wedge w_2 \wedge w_3$ and $\bar{I} = \bar{w}_1 \wedge \bar{w}_2 \wedge \bar{w}_3$ which satisfy $I\bar{I} = 1$.

The Lie algebra $sl(4)$ of the 3D projective transformations is isomorphic with the bivector algebra of $G_{3,3}$. The canonical homomorphism from $SL(4)$ to $Spin(3,3)$ is not Surjective, and the projective transformations of negative determinant do not induce orthogonal transformations in the Plücker coordinate space of lines. To overcome these Limitations, Hongbo Li et al. [7] proposed the group called $Pin^{sp}(3,3)$ with $Pin(3,3)$ as its normal subgroup, in order to quadruple-cover the group of the projective transformations and polarities.

Let us review some geometric algebras for the representation of points, lines, planes, and their associated Lie groups. The conformal geometric algebra $G_{4,1}$ of the 3D space [8] is equipped with $\begin{bmatrix} 5 \\ 2 \end{bmatrix} = 10$ bivectors which as Lie algebra generators

suffice to build the conformal group consisting of $SE(3)$, dilations and inversions. In addition, the motor algebra $G^{+}_{3,0,1}$ [8, 9] with $\begin{bmatrix} 4 \\ 2 \end{bmatrix} = 6$ bivector basis

$$\{e_{01}, e_{02}, e_{03}, e_{23}, e_{31}, e_{12}\} \tag{3.152}$$

correspond to the *Plücker* coordinates of the lines. Now, given two homogeneous points $P, Q \in G_{3,0,1}$, we compute a line wedging both points

$$\boldsymbol{L} = P \wedge Q = P \wedge n = (e_0 + \boldsymbol{p}) \wedge \boldsymbol{n} = e_0 \wedge \boldsymbol{n} + \boldsymbol{p} \wedge \boldsymbol{n}, \tag{3.153}$$

where $\boldsymbol{n} = \boldsymbol{P} - \boldsymbol{Q}$ is the line orientation and $\boldsymbol{m} = \boldsymbol{p} \wedge \boldsymbol{n}$ is its moment.

In the space spanned by the bivectors of Eq. (3.152), not all bivectors will represent lines; only lines $\boldsymbol{L} = l_{01}e_{01} + l_{02}e_{02} + l_{03}e_{03} + l_{23}e_{23} + l_{31}e_{31} + l_{12}e_{12}$ fulfill the so-called Grassmann–Plücker relation

$$M^4_2 : l_{01}l_{23} + l_{02}l_{31} + l_{03}l_{12} = 0. \tag{3.154}$$

This relation defines a quadric called the manifold of Study [10] which is an algebraic variety M^4_2 of degree 2 and dimension 4. Hence, $G^{+}_{3,0,1}$ is appropriate for line geometry and handle $SE(3)$, and it can be used to model the kinematics of points, lines, and planes, as shown in [8, 9]. As you see to model the Lie group of the projective geometry $SL(4)$, we require more bivectors; that is why we consider $G_{3,3}$ which offers $\begin{bmatrix} 6 \\ 2 \end{bmatrix} = 15$ bivectors enough to handle the 15-DOF of the projective transformation of $\mathbb{R}^4$. Thus, we can infer that the points and planes of the geometric algebra $G_{3,0,1}$ for projective geometry and its group action $GL(4)$ upon $\mathbb{R}^4$ can be modeled in terms of null 3-vector representations and the adjoint action of $Pin^{sp}(3,3)$ as we will explain next in much more detail.

3.7.1 Line Geometry Using Null Geometric Algebra Over $\mathbb{R}^{3,3}$

Let e_0, e_1, e_2, e_3 be an orthonormal basis of $\mathbb{R}^4$ which expands the affine plane, where the null vector $e_4 \in G_{3,0,1}$ called e_0 is the origin of the Euclidean affine space 3D space $\mathbb{E}^3$. The unit vectors e_1, e_2, e_3 are points at the plane at infinity $\langle e_1, e_2, e_3 \rangle$, and they represent the directions of three basis projective lines $\{e_{23}, e_{31}, e_{12}\}$. These lines are in turn the intersections (meets) of three planes $\{e_{023}, e_{012}, e_{012}\}$ which cross the origin e_0.

We project the affine plane onto $\mathbb{R}^{3,3} = N_3 \oplus \overline{N}_3$; hence, the basis e_0, e_1, e_2, e_3 induces a Witt basis.

Li and Zhang [7, 11] proposed a new model of 3D projective geometry by taking the null vectors of $\mathbb{R}^{3,3}$ as algebraic generators. They defined points and planes as two connected components of the set of null 3-spaces of $\mathbb{R}^{3,3}$, respectively, and showed that whenever an element of $Spin_o(3,3)$ acts upon $\mathbb{R}^{3,3}$, it induces a projective transformation by means the outermorphism of the action upon the null 3-vectors representing 3D points and planes. Later on, Klawitter [12] along the same lines of thought, he proposed an explicit expression of the spinor inducing a projective transformation in 4×4 matrix form.

Embedding the line basis of Eq. (3.152) via the function

$$\wp : \bigwedge^{2}(\mathbb{R}^4) \rightarrow \mathbb{R}^{3,3}, \tag{3.155}$$

we get a space spanned by a null basis given by

$$\{w_1, w_2, w_3, \overline{w}_1, \overline{w}_2, \overline{w}_3\}. \tag{3.156}$$

The embedding of lines $\boldsymbol{L} = e_0 \wedge \boldsymbol{n} + \boldsymbol{p} \wedge \boldsymbol{n} \in G_{3,1}$ of $\mathbb{R}^4$ via the function $\wp(\cdot)$ in $\mathbb{R}^{3,3}$ yields

$$\wp : \boldsymbol{L} \longrightarrow n_x w_1 + n_y w_2 + n_z w_3 + m_x \overline{w}_1 + m_y \overline{w}_2 + m_z \overline{w}_3. \tag{3.157}$$

Note in Eq. (5.2), we model a line in G_3 of $\mathbb{R}^3$ using the available k-blades as

$$\boldsymbol{L} = \mathbf{n} + \boldsymbol{m}, \tag{3.158}$$

where $\mathbf{n}$ and $\boldsymbol{m}$ are a vector and a bivector. In motor algebra $G^{+}_{3,0,1}$, a line is represented in terms of the bivectors $\boldsymbol{n}$ and $\boldsymbol{m}$ (expanded in the dual bivector basis) and the pseudoscalar I which squares to null,

$$\boldsymbol{L} = \boldsymbol{n} + \boldsymbol{m} = \boldsymbol{n} + I\boldsymbol{m}. \tag{3.159}$$

Finally in $G_{3,3}$ of $\mathbb{R}^{3,3}$, we represent the line in terms of null vectors as given in Eq. (3.157).

3.7.2 *Projective Transformations Using Null Geometric Algebra Over* $\mathbb{R}^{3,3}$

Recently, Dorst [13] proposed bivector generators for 3D projective transformations. In this subsection, we will explain using our previous equations how we derive the affine and projective transformations in terms of versors of $G_{3,3}$. For that, we have to embed the projective transformation of $\mathbb{R}^4$ into $\mathbb{R}^{3,3}$ and use a 6×6 bivector matrix

similar as in Eq. (3.66) and then select the bivectors generators of $\mathbb{R}^{3,3}$ to construct specialized versors for translation, rotation, perspectivity, and affine transformations. Firstly, let us to discuss the inner product and the isometry between $\mathbb{R}^4$ and $\mathbb{R}^{3,3}$.

For the metric space of oriented lines in $\mathbb{R}^{3,3}$, we define next an inner product via the *bracket* of $G_{3,1}$ in $\mathbb{R}^4$ a 4D-volume, given $L_1, L_2 \in G_{3,0,1}$ and the lines $l_1, l_2 \in G_{3,3}$

$$L_1 \cdot L_2 = (P_1 \wedge Q_1) \cdot (P_2 \wedge Q_2) = [\wp^{-1}(l_1) \wedge \wp^{-1}(l_2)] = [P_1, Q_1, P_2, Q_2],$$

the inner product of L_1, L_2 gives the volume spanned by the line orientations and their orthogonal distance or fut, when they intersect the bracket or volume is zero.

Consider a transformation $f : \mathbb{R}^4 \longrightarrow \mathbb{R}^4$ for points and $\underline{f}$ its outermorphism on $\bigwedge^2 \mathbb{R}^4$, the corresponding transformation $F : \mathbb{R}^{3,3} \to \mathbb{R}^{3,3}$ is $F() \equiv \wp(\underline{f})(\wp^{-1}())$ on lines and screws. Let us compute the dot product of two lines

$$\begin{aligned} F(l_1) \cdot F(l_2) &= F(\wp(L_1)) \cdot F(\wp(L_2)) = \wp(\underline{f}(L_1)) \cdot \wp(\underline{f}(L_2)), \\ &= [\underline{f}(L_1) \wedge \underline{f}(L_2)] = [\underline{f}(L_1 \wedge L_2)] = det(f)[L_1 \wedge L_2], \\ &= det(f)\wp(L_1) \cdot \wp(L_2) = det(f) l_1 \cdot l_2. \end{aligned} \tag{3.160}$$

By the case of an orthogonal transformation $det(f) = 1$ and $det(F) = det(f)^3 = 1$. Projective transformations with $det(f) = -1$ do not induce orthogonal transformations in the Pluckër space of lines.

For the applications as in computer vision and graphics engineering, it is needed projective transformations, let us find the projective transformations of lines. In the projective space $\mathbb{P}^3$, given the orientation $\mathbf{u} \in \mathbb{R}^3$ and a point $\mathbf{x} \in \mathbb{R}^3$ touching the line, a line $\mathbf{L}$ is computed as follows:

$$\mathbf{L} = \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix} \wedge \begin{bmatrix} \mathbf{n} \\ 0 \end{bmatrix} = \begin{bmatrix} \mathbf{n} \\ \mathbf{x} \times \mathbf{n} \end{bmatrix} = \begin{bmatrix} \mathbf{n} \\ \mathbf{m} \end{bmatrix}, \tag{3.161}$$

where $\mathbf{m}$ stands for the moment of the line, which is a vector normal to the surface where the line lies; note that this line belongs to the projective space $\mathbb{P}^5$. An homography $\mathtt{H}$ in $\mathbb{P}^3$ is given by

$$\mathtt{H} = \begin{bmatrix} \mathtt{A} & \mathbf{t} \\ \mathbf{f} & h \end{bmatrix}, \tag{3.162}$$

where $\mathtt{A}$ is an affine transformation matrix, $\mathbf{t}$ is a translation vector, and if $h = 1$ the vector $\mathbf{f}$ corresponds to the perspectivity in the pinhole model of a digital camera [14]. If we apply the homography the two projective points and compute the wedge, we obtain the line after the action of the homography

$$\begin{aligned}
\mathbf{L}' &= \mathtt{H}\begin{bmatrix}\mathbf{x}\\1\end{bmatrix} \wedge \mathtt{H}\begin{bmatrix}\mathbf{n}\\0\end{bmatrix} = \begin{bmatrix}\mathtt{A}\mathbf{x}+\mathbf{t}\\ \mathbf{f}^T\mathbf{x}+h\end{bmatrix} \wedge \begin{bmatrix}\mathtt{A}\mathbf{n}\\ \mathbf{f}^T\mathbf{n}\end{bmatrix},\\
&= \begin{bmatrix}\mathtt{A}\mathbf{n}(\mathbf{f}^\mathbf{T}\mathbf{x}+\mathbf{h}) - (\mathtt{A}\mathbf{x}+\mathbf{t})\mathbf{f}^\mathbf{T}\mathbf{n}\\ (\mathtt{A}\mathbf{x}+\mathtt{t})\times\mathtt{A}\mathbf{n}\end{bmatrix},\\
&= \begin{bmatrix}h\mathtt{A}\mathbf{n} + (\mathbf{f}^\mathbf{T}\mathbf{x})\mathtt{A}\mathbf{n} - \mathtt{A}\mathbf{x}(\mathbf{f}^\mathbf{T}\mathbf{n}) - \mathbf{t}(\mathbf{f}^\mathbf{T}\mathbf{n})\\ \mathtt{A}\mathbf{x}\times\mathtt{A}\mathbf{n} + \mathbf{t}\times\mathtt{A}\mathbf{n}\end{bmatrix},\\
&= \begin{bmatrix}(h\mathtt{A} - \mathbf{t}^T\mathbf{f}) & -\mathtt{A}[\mathbf{f}]_\times\\ [\mathbf{t}]_\times\mathtt{A} & (\det\mathtt{A})\mathtt{A}^{-T}\end{bmatrix}\begin{bmatrix}\mathbf{n}\\ \mathbf{x}\times\mathbf{n}\end{bmatrix} = \mathtt{T}(\mathbf{L}),
\end{aligned} \tag{3.163}$$

where the vector triple product was used in $(\mathbf{f}\cdot\mathbf{x})\mathtt{A}\mathbf{n} - \mathtt{A}\mathbf{x}(\mathbf{f}\cdot\mathbf{n}) = \mathtt{A}(\mathbf{f}\cdot\mathbf{x})\mathbf{n} - (\mathbf{f}\cdot\mathbf{n})\mathtt{A}\mathbf{x} = \mathtt{A}(-\mathbf{x}(\mathbf{f}\cdot\mathbf{n}) + \mathbf{n}(\mathbf{f}\cdot\mathbf{x})) = \mathtt{A}((\mathbf{f}\cdot\mathbf{x})\mathbf{n} - (\mathbf{f}\cdot\mathbf{n})\mathbf{x})) = \mathtt{A}[\mathbf{f}\times\mathbf{n}\times\mathbf{x}] = -\mathtt{A}[\mathbf{f}\times\mathbf{x}\times\mathbf{n}] = -\mathtt{A}[\mathbf{f}]\times(\mathbf{x}\times\mathbf{n})$, where $[\mathbf{f}]_\times$ stands for the operator as an antisymmetric matrix which represents the cross product with $\mathbf{f}$.

3.7.3 Lie Groups in $\mathbb{R}^{3,3}$

Equation (3.163) represents the line transformation into the projective space $\mathbb{P}^5$,

$$\mathtt{T} = \begin{bmatrix}(h\mathtt{A} - \mathbf{t}^T\mathbf{f}) & -\mathtt{A}[\mathbf{f}]_\times\\ [\mathbf{t}]_\times\mathtt{A} & (\det\mathtt{A})\mathtt{A}^{-T}\end{bmatrix}, \tag{3.164}$$

which will be used to find the generators of its Lie algebra. The basis to expand the Lie operators of $\mathtt{T}$ in $\mathbb{R}^{3,3}$ is given by the unit basis $e_i, \bar{e}_i$ with $e_i\cdot e_j = \delta_{i,j}$ and $\bar{e}_i\cdot\bar{e}_j = -\delta_{i,j}$ and the null basis $w_i, \bar{w}_i$ with $w_i\cdot\bar{w}_j = \delta_{i,j}$. The inner product on the unit basis gives the signature of the space in $\mathbb{R}^{3,3}$.

Similarly, one obtains the inner product table using the null basis, as shown in the tables below:

$\cdot$	e_1	e_2	e_3	$\bar{e}_1$	$\bar{e}_2$	$\bar{e}_3$
e_1	1	0	0	0	0	0
e_2	0	1	0	0	0	0
e_3	0	0	1	0	0	0
$\bar{e}_1$	0	0	0	−1	0	0
$\bar{e}_2$	0	0	0	0	−1	0
$\bar{e}_3$	0	0	0	0	0	−1

$\cdot$	w_1	w_2	w_3	$\bar{w}_1$	$\bar{w}_2$	$\bar{w}_3$
w_1	0	0	0	1	0	0
w_2	0	0	0	0	1	0
w_3	0	0	0	0	0	1
$\bar{w}_1$	1	0	0	0	0	0
$\bar{w}_2$	0	1	0	0	0	0
$\bar{w}_3$	0	0	1	0	0	0

The projective transformation 3.164 is represented using the unit basis or the null basis as follows:

$$\mathtt{T}_{e_i,\bar{e}_j} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & 0 & -f_3 & f_2 \\ a_{21} & a_{22} & a_{23} & f_3 & 0 & -f_1 \\ a_{31} & a_{32} & a_{33} & -f_2 & f_1 & 0 \\ 0 & t_3 & -t_2 & -a_{11} & -a_{21} & -a_{31} \\ -t_3 & 0 & t_1 & -a_{12} & -a_{22} & -a_{32} \\ t_2 & -t_1 & 0 & -a_{13} & -a_{23} & -a_{33} \end{bmatrix}, \tag{3.165}$$

$$\mathtt{T}_{w_i,\bar{w}_j} = \begin{bmatrix} 0 & -f_3 & f_2 & -a_{11} & -a_{21} & -a_{31} \\ f_3 & 0 & -f_1 & -a_{12} & -a_{22} & -a_{32} \\ -f_2 & f_1 & 0 & -a_{13} & -a_{23} & -a_{33} \\ a_{11} & a_{12} & a_{13} & 0 & t_3 & -t_2 \\ a_{21} & a_{22} & a_{23} & -t_3 & 0 & t_1 \\ a_{31} & a_{32} & a_{33} & t_2 & -t_1 & 0 \end{bmatrix}. \tag{3.166}$$

The projective transformation of a vector $\boldsymbol{x} \in \mathbb{R}^{3,3}$ can be computed using the Lie group versorsindexversors! Lie group and the generators of Lie algebra in terms of bivectors, where the generators can be expanded as linear combinations of different bivectors, namely,

$$\boldsymbol{x}' = \boldsymbol{V}(\boldsymbol{x})\widetilde{\boldsymbol{V}} = e^{\boldsymbol{B}}\boldsymbol{x}e^{-\boldsymbol{B}} = e^{\sum_{i,j} t_{ij}\boldsymbol{B}_{i,j}}\boldsymbol{x}e^{-\sum_{i,j} t_{ij}\boldsymbol{B}_{i,j}}, \tag{3.167}$$

where $t_{ij} \in \mathbb{R}$ are computed based on the entries of the transformation matrix of Eq. (3.165) or of Eq. (3.166), respectively. The bivectors $\boldsymbol{B}_{i,j}$ are computed in terms of the unit basis $e_i \wedge e_j$ or null basis $w_i \wedge w_j$ as shown below in Eqs. (3.168)–(3.169).

In order to compute any Lie generators in terms of bivectors, firstly one takes the derivatives of $\mathtt{T}_{e_i,\bar{e}_j}$ Eq. (3.165) or $\mathtt{T}_{w_i,\bar{w}_j}$ Eq. (3.166) with respect to a specific parameter ν (degree of freedom), and then one evaluates at $\nu = 0$ to get the corresponding generator of certain Lie algebra associated to the projective Lie group $SL(4,\mathbb{R})$. Then, one expresses the resulting Lie generators using bivectors corresponding to the transformation of Eq. (3.165) or of Eq. (3.166). Similar as the procedure given in Eqs. (3.63) and (3.66), the resulting Lie generator or matrix in terms of unit vector bivectors $e_i \wedge \bar{e}_j$ reads

$$\mathbf{T}_{e_i\wedge\bar{e}_j} = \begin{bmatrix} 0 & a_{12}e_1\wedge e_2 & a_{13}e_1\wedge e_3 & 0 & -f_3e_1\wedge\bar{e}_2 & f_2e_1\wedge\bar{e}_3 \\ a_{21}e_2\wedge e_1 & 0 & a_{23}e_2\wedge e_3 & f_3e_2\wedge\bar{e}_1 & 0 & -f_1e_2\wedge\bar{e}_3 \\ a_{31}e_3\wedge e_1 & a_{32}e_3\wedge e_2 & 0 & -f_2e_3\wedge\bar{e}_1 & f_1e_3\wedge\bar{e}_2 & 0 \\ 0 & t_3\bar{e}_1\wedge e_2 & -t_2\bar{e}_1\wedge e_3 & 0 & -a_{21}\bar{e}_1\wedge\bar{e}_2 & -a_{31}\bar{e}_1\wedge\bar{e}_3 \\ -t_3\bar{e}_2\wedge e_1 & 0 & t_1\bar{e}_2\wedge e_3 & -a_{12}\bar{e}_2\wedge\bar{e}_1 & 0 & -a_{32}\bar{e}_2\wedge\bar{e}_3 \\ t_2\bar{e}_3\wedge e_1 & -t_1\bar{e}_3\wedge e_2 & 0 & -a_{13}\bar{e}_3\wedge\bar{e}_1 & -a_{23}\bar{e}_3\wedge\bar{e}_2 & 0 \end{bmatrix} \tag{3.168}$$

or in terms of null vector bivectors $w_i \wedge \bar{w}_j$

$$\mathbf{T}_{w_i \wedge \bar{w}_j} = \begin{bmatrix} 0 & -f_3 w_1 \wedge w_2 & f_2 w_1 \wedge w_3 & -s_1 w_1 \wedge \bar{w}_1 & -a_{21} w_1 \wedge \bar{w}_2 & -a_{31} w_1 \wedge \bar{w}_3 \\ f_3 w_2 \wedge w_1 & 0 & -f_1 w_2 \wedge w_3 & -a_{12} w_2 \wedge \bar{w}_1 & -s_2 w_2 \wedge \bar{w}_2 & -a_{32} w_2 \wedge \bar{w}_3 \\ -f_2 w_3 \wedge w_1 & f 1 w_3 \wedge w_2 & 0 & -a_{13} w_3 \wedge \bar{w}_1 & -a_{23} w_3 \wedge \bar{w}_2 & -s_3 w_3 \wedge \bar{w}_3 \\ s_1 \bar{w}_1 \wedge w_1 & a_{12} \bar{w}_1 \wedge w_2 & a_{13} \bar{w}_1 \wedge w_3 & 0 & t_3 \bar{w}_1 \wedge \bar{w}_2 & -t_2 \bar{w}_1 \wedge \bar{w}_3 \\ a_{21} \bar{w}_2 \wedge w_1 & s_2 \bar{w}_2 \wedge w_2 & a_{23} \bar{w}_2 \wedge w_3 & -t_3 \bar{w}_2 \wedge \bar{w}_1 & 0 & t_1 \bar{w}_2 \wedge \bar{w}_3 \\ a_{31} \bar{w}_3 \wedge w_1 & a_{32} \bar{w}_3 \wedge w_2 & s_3 \bar{w}_3 \wedge w_3 & t_2 \bar{w}_3 \wedge \bar{w}_1 & -t_1 \bar{w}_3 \wedge \bar{w}_2 & 0 \end{bmatrix}. \tag{3.169}$$

Let us start computing the Lie generators for the translation.

$$\mathcal{L}_\nu = \frac{\mathrm{d}\mathrm{T}_{w_i, \bar{w}_j \nu}}{\mathrm{d}\nu}|_{\nu \to 0}. \tag{3.170}$$

Translation: Taking the derivatives for each degree of freedom $\{\mathtt{t}_x, \mathtt{t}_y, \mathtt{t}_z\}$ of

$$\mathrm{T}_\mathtt{t} = \begin{bmatrix} \mathtt{I} & \mathtt{O} \\ [\mathbf{t}]_\times & \mathtt{O} \end{bmatrix}, \tag{3.171}$$

$$\mathcal{L}_{\mathtt{t}_x} = \frac{\mathrm{d}T_{w_i, \bar{w}_j \mathtt{t}_x}}{\mathrm{d}\mathtt{t}_x}|_{\mathtt{t}_x \to 0} = \begin{bmatrix} & \mathtt{I} & & \mathtt{O} \\ \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix} & & & \\ & & & \mathtt{O} \end{bmatrix}, \tag{3.172}$$

similarly for $\{\mathtt{t}_y, \mathtt{t}_z\}$, gathering these results we obtain the Lie operator for translation

$$\mathcal{L}_\mathtt{t} = \begin{bmatrix} & \mathtt{I} & & \mathtt{O} \\ \begin{bmatrix} 0 & 1 & -1 \\ -1 & 0 & 1 \\ 1 & -1 & 0 \end{bmatrix} & & & \\ & & & \mathtt{I} \end{bmatrix}. \tag{3.173}$$

The versor for translation called *translator* can be computed either using the bivectors $e_i \wedge \bar{e}_j = e_i \bar{e}_j$ (orthonormal basis) of Eq. (3.168) or following the similar approach using the bivectors $w_i \wedge \bar{w}_j$ of Eq. (3.169) to get

$$\begin{aligned} \boldsymbol{T} &= e^{\boldsymbol{B}_\mathbf{t}} = e^{\frac{t_3 \bar{e}_1 e_2 - t_2 \bar{e}_1 e_3 + t_1 \bar{e}_2 e_3}{2}}, \qquad (3.174) \\ &= e^{\frac{t_3 \bar{w}_1 \wedge \bar{w}_2 - t_2 \bar{w}_1 \wedge \bar{w}_3 + t_1 \bar{w}_2 \wedge \bar{w}_3}{2}}, \\ &= 1 + \frac{1}{2}(t_3 \bar{e}_1 e_2 - t_2 \bar{e}_1 e_3 + t_1 \bar{e}_2 e_3), \\ &= 1 + \frac{1}{2}(t_3 \bar{w}_1 \wedge w_2 - t_2 \bar{w}_1 \wedge \bar{w}_3 + t_1 \bar{w}_2 \wedge \bar{w}_3). \qquad (3.175) \end{aligned}$$

Perspectivity: Taking the derivatives for each degree of freedom $\{\mathtt{f}_x, \mathtt{f}_y, \mathtt{f}_z\}$ of

$$\mathtt{T}_f = \begin{bmatrix} \mathtt{I} & [\mathbf{f}]_\times \\ \mathtt{O} & \mathtt{I} \end{bmatrix}, \tag{3.176}$$

$$\mathcal{L}_{\mathtt{f}_x} = \frac{\mathtt{d}T_{\mathtt{f}_x}}{\mathtt{d}\mathtt{f}_x}|_{\mathtt{f}_x \to 0} = \begin{bmatrix} & \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix} \\ \mathtt{I} & \\ \mathtt{O} & \mathtt{I} \end{bmatrix}, \tag{3.177}$$

similarly for $\{\mathtt{f}_y, \mathtt{f}_z\}$, gathering these results we obtain the Lie operator for perspectivity

$$\mathcal{L}_{\mathtt{f}} = \begin{bmatrix} & \begin{bmatrix} 0 & -1 & 1 \\ 1 & 0 & -1 \\ -1 & 1 & 0 \end{bmatrix} \\ \mathtt{I} & \\ \mathtt{O} & \mathtt{I} \end{bmatrix}. \tag{3.178}$$

The versor for perspectivity called *perspector* can be computed either using the bivectors $e_i\bar{e}_j$ of Eq. (3.168) or following the similar approach using the bivectors $w_i \wedge \bar{w}_j$ of Eq. (3.169) to get

$$\begin{aligned} \boldsymbol{P} &= e^{\boldsymbol{B}f} = e^{\frac{-f_3 e_1\bar{e}_2 + f_2 e_1\bar{e}_3 - f_1 e_2\bar{e}_3}{2}}, \\ &= e^{\frac{-f_3 w_1 \wedge w_2 + f_2 w_1 \wedge w_3 - f_1 w_2 \wedge w_3}{2}}, \\ &= 1 + \frac{1}{2}(-f_3 e_1\bar{e}_2 + f_2 e_1\bar{e}_3 - f_1 e_2\bar{e}_3), \\ &= 1 + \frac{1}{2}(-f_3 w_1 \wedge w_2 + f_2 w_1 \wedge w_3 - f_1 w_2 \wedge w_3). \end{aligned} \tag{3.179}$$

Rotation: Taking the derivatives for each Euler angle $\{\theta, \phi, \psi\}$ of

$$\mathtt{T}_{\mathtt{R}} = \begin{bmatrix} \mathtt{R} & \mathtt{O} \\ \mathtt{O} & \mathtt{R}^{-\mathtt{T}} \end{bmatrix}, \tag{3.180}$$

we obtain first the Lie operator for the 3D rotation $\mathtt{R}_z$ with respect to the axis z

$$\mathcal{L}_\theta = \frac{\mathtt{d}\mathtt{T}_{\mathtt{R}_z}}{\mathtt{d}\theta}|_{\theta \to 0} = \frac{\mathtt{d}\begin{bmatrix} \mathtt{R}_z & \mathtt{O} \\ \mathtt{O} & \mathtt{R}_z^T \end{bmatrix}}{\mathtt{d}\theta}|_{\theta \to 0}, \tag{3.181}$$

$$
\begin{aligned}
&= \frac{\mathtt{d}\begin{bmatrix} \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} & \mathtt{O} \\ \mathtt{O} & \begin{bmatrix} \cos(\theta) & \sin(\theta) & 0 \\ -\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \end{bmatrix}}{\mathtt{d}\theta}|_{\theta\to 0} \\
&= \begin{bmatrix} \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} & \mathtt{O} \\ \mathtt{O} & \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \end{bmatrix}
\end{aligned} \tag{3.182}
$$

and similarly for those angles ϕ, ψ with respect to axes y and x, gathering these results we obtain the Lie operator for the rotations $\mathtt{R} = \mathtt{R}_\mathtt{z}\mathtt{R}_\mathtt{y}\mathtt{R}_\mathtt{z}$

$$
\mathcal{L}_\mathtt{R} = \begin{bmatrix} \begin{bmatrix} 0 & -1 & -1 \\ 1 & 0 & -1 \\ 1 & 1 & 0 \end{bmatrix} & \mathtt{O} \\ \mathtt{O} & \begin{bmatrix} 0 & 1 & 1 \\ -1 & 0 & 1 \\ -1 & -1 & 0 \end{bmatrix} \end{bmatrix}. \tag{3.183}
$$

The versor for rotation called *rotator* can be computed either using the bivectors $e_i\bar{e}_j$ of Eq. (3.168) or following the similar approach using the bivectors $w_i \wedge \bar{w}_j$ of Eq. (3.169) to get

$$
\begin{aligned}
\boldsymbol{R} &= e^{\boldsymbol{B}_R}, \\
&= e^{\frac{(-e_2e_3+\bar{e}_2\bar{e}_3)+(-e_1e_3+\bar{e}_1\bar{e}_3)+(-e_1e_2+\bar{e}_1\bar{e}_2)}{2}}, \\
&= e^{\frac{(w_2\wedge\bar{w}_3-\bar{w}_2\wedge w_3)+(w_1\wedge\bar{w}_3-\bar{w}_1\wedge w_3)+(w_1\wedge\bar{w}_2-\bar{w}_1\wedge w_2)}{2}}.
\end{aligned} \tag{3.184}
$$

Lorentz transformation: Taking the derivatives for each variable $\{\vartheta, \varphi, \varpi\}$ of

$$
\mathtt{T}_\mathtt{LO} = \begin{bmatrix} \mathtt{LO} & \mathtt{O} \\ \mathtt{O} & -\mathtt{LO}^{-\mathtt{T}} \end{bmatrix}, \tag{3.185}
$$

we obtain first the Lie operator for the 3D Lorentz transformation $\mathtt{LO}$ with respect to the axis z and variable ϑ

$$
\mathcal{L}_{\mathtt{LO}_\vartheta} = \frac{\mathtt{dLO}_\vartheta}{\mathtt{d}\vartheta}|_{\vartheta\to 0} = \frac{\mathtt{d}\begin{bmatrix} \mathtt{LO}_\vartheta & \mathtt{O} \\ \mathtt{O} & -\mathtt{LO}_\vartheta^T \end{bmatrix}}{\mathtt{d}\theta}|_{\theta\to 0}, \tag{3.186}
$$

$$
\begin{aligned}
&= \frac{\mathrm{d}\begin{bmatrix} \begin{bmatrix} \cosh(\vartheta) & \sinh(\vartheta) & 0 \\ \sinh(\vartheta) & \cosh(\vartheta) & 0 \\ 0 & 0 & 1 \end{bmatrix} & \mathbb{O} \\ \mathbb{O} & \begin{bmatrix} \cosh(\vartheta) & -\sinh(\vartheta) & 0 \\ -\sinh(\vartheta) & \cosh(\vartheta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \end{bmatrix}}{\mathrm{d}\vartheta}\Big|_{\vartheta\to 0} \\
&= \begin{bmatrix} \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} & \mathbb{O} \\ \mathbb{O} & \begin{bmatrix} 0 & -1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \end{bmatrix}
\end{aligned} \tag{3.187}
$$

and similarly for the variables $\{\varphi, \varpi\}$ with respect to axes y and x, gathering these results we obtain the Lie operator for the Lorentz transformation involving the variables $\{\vartheta, \varphi, \varpi\}$

$$
\mathcal{L}_{\mathrm{LO}} = \begin{bmatrix} \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix} & \mathbb{O} \\ \mathbb{O} & \begin{bmatrix} 0 & -1 & -1 \\ -1 & 0 & -1 \\ -1 & -1 & 0 \end{bmatrix} \end{bmatrix}. \tag{3.188}
$$

The versor for the Lorentz transformation called *Lorentor* can be computed either using the bivectors $e_i\bar{e}_j$ of Eq. (3.168) or following the similar approach using the bivectors $w_i \wedge \bar{w}_j$ of Eq. (3.169) to get

$$
\begin{aligned}
\boldsymbol{R}_L &= e^{\boldsymbol{B}_L}, \\
&= e^{\frac{(e_2\wedge e_3 - \bar{e}_2\wedge\bar{e}_3)+(e_1\wedge e_3 - \bar{e}_1\wedge\bar{e}_3)+(e_1\wedge e_2 - \bar{e}_1\wedge\bar{e}_2)}{2}}, \\
&= e^{\frac{(\bar{w}_2\wedge w_3 - w_2\wedge\bar{w}_3)+(\bar{w}_1\wedge w_3 - w_1\wedge\bar{w}_3)+(\bar{w}_1\wedge w_2 - w_1\wedge\bar{w}_2)}{2}}.
\end{aligned} \tag{3.189}
$$

The next Lie operators will be found using the parameters of the affine transformation matrix $\mathtt{A}$ in the transformation $\mathtt{T}$ of Eq. (3.164) with $h = 1$ and $\det \mathtt{A} = 1$ or equivalently Eq. (3.165)

$$
\mathtt{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}. \tag{3.190}
$$

Shear: Taking the derivatives for each of the variables off the diagonal of $\mathtt{A}$ $\{a_{12}, a_{13}, a_{21}, a_{23}, a_{31}, a_{32}\}$ in

$$\mathtt{T}_{\mathtt{A}} = \begin{bmatrix} \mathtt{A} & \mathtt{O} \\ \mathtt{O} & -\mathtt{A}^{-T} \end{bmatrix}, \tag{3.191}$$

we obtain first the Lie operator for a partially shear $\mathtt{A}_{a_{23}}$

$$\mathcal{L}_{a_{23}} = \frac{\mathtt{d}\mathtt{T}_{A_1}}{\mathtt{d}a_{23}}|_{a_{23}\to 0} = \frac{\mathtt{d}\begin{bmatrix} \mathtt{A}_1 & \mathtt{O} \\ \mathtt{O} & -\mathtt{A}_1^T \end{bmatrix}}{\mathtt{d}a_{23}}|_{a_{23}\to 0} \tag{3.192}$$

$$= \frac{\mathtt{d}\begin{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & a_{23} \\ 0 & 0 & 1 \end{bmatrix} & \mathtt{O} \\ \mathtt{O} & \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -a_{23} & 1 \end{bmatrix} \end{bmatrix}}{\mathtt{d}a_{23}}|_{a_{23}\to 0}$$

$$= \begin{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} & \mathtt{O} \\ \mathtt{O} & \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & -1 & 0 \end{bmatrix} \end{bmatrix} \tag{3.193}$$

and similarly for the variables $\{a_{13}, a_{21}, a_{23}, a_{31}, a_{32}\}$; gathering these results, we obtain the Lie operator for the shears

$$\mathcal{L}_{\mathtt{A}} = \begin{bmatrix} \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix} & \mathtt{O} \\ \mathtt{O} & \begin{bmatrix} 0 & -1 & -1 \\ -1 & 0 & -1 \\ -1 & -1 & 0 \end{bmatrix} \end{bmatrix}. \tag{3.194}$$

If we combine the Lie operators of opposite shears of $\{a_{12}, a_{13}, a_{21}, a_{23}, a_{31}, a_{32}\}$, we obtain the Lie generators to squeeze con respect to the axes z, y, and x, respectively. The versor for the Shear transformation called *Shearotor* can be computed either using the bivectors $e_i\bar{e}_j$ of Eq. (3.168) or following the similar approach using the bivectors $w_i \wedge \bar{w}_j$ of Eq. (3.169) to get

$$\begin{aligned} \boldsymbol{S} &= e^{\boldsymbol{B}_S} \\ &= e^{\frac{a_{23}(e_2\wedge e_3 - \bar{e}_3\wedge\bar{e}_2)+a_{13}(e_1\wedge e_3-\bar{e}_3\wedge\bar{e}_1)+a_{12}(e_1\wedge e_2-\bar{e}_2\wedge\bar{e}_1)}{2}} \\ &\quad e^{\frac{a_{32}(e_3\wedge e_2-\bar{e}_2\wedge\bar{e}_3)+a_{31}(e_3\wedge e_1-\bar{e}_1\wedge\bar{e}_3)+a_{21}(e_2\wedge e_1-\bar{e}_1\wedge\bar{e}_2)}{2}} \\ &= e^{\frac{a_{23}(\bar{w}_2\wedge w_3 - w_3\wedge\bar{w}_2)+a_{13}(\bar{w}_1\wedge w_3-w_3\bar{w}_1)+a_{12}(\bar{w}_1\wedge w_2-w_2\wedge\bar{w}_1)}{2}} \\ &\quad e^{\frac{a_{32}(\bar{w}_3\wedge w_2-w_2\wedge\bar{w}_3)+a_{31}(\bar{w}_3\wedge w_1-w_1\bar{w}_3)+a_{21}(\bar{w}_2\wedge w_1-w_1\wedge\bar{w}_2)}{2}}. \end{aligned} \tag{3.195}$$

Dilation: The Lie operators for scaling or dilation will be found by taking the derivative with respect to the parameters of the dilation transformation matrix $\mathtt{D}$ in the transformation $\mathtt{T}$ of Eq. (3.164)

$$\mathtt{D} = \begin{bmatrix} \{a_{11}, e^{\pm s_1}\} & 0 & 0 \\ 0 & \{a_{22}, e^{\pm s_2}\} & 0 \\ 0 & 0 & \{a_{33}, e^{\pm s_3}\} \end{bmatrix}, \tag{3.196}$$

where you can represent certain dilation transformation in terms of the diagonal entries either $a_{i,i}$ or $e^{\pm s_i}$ $(i = 1, \ldots, 3)$. Now, for the Lie operator for a directional scaling by a_{11} or e^{s_1} along axis x, we take the derivative with respect the parameter a_{11} or s_1

$$\begin{aligned}
\mathcal{L}_{\mathcal{D}_{\{a_{11},s_1\}}} &= \frac{\mathtt{d}\mathtt{T}_{\mathcal{D}_{\{a_{11},s_1\}}}}{\mathtt{d}\{a_{11}, s_1\}}|_{\{a_{11},s_1\}} \to 0 \\
&= \frac{\mathtt{d}\begin{bmatrix} \begin{bmatrix} a_{11}, e^{s_1} & 0 & 0 \\ 0 & -a_{22}, e^{-s_1} & 0 \\ 0 & 0 & -a_{33}, e^{-s_1} \end{bmatrix} & 0 \\ 0 & \mathcal{D}_{\{-a_{11},-s_1\}} \end{bmatrix}}{\mathtt{d}\{a_{11}, s_1\}}|_{\{a_{11},s_1\}} \to 0 \\
&= \frac{\mathtt{d}\begin{bmatrix} \mathcal{D}_{\{a_{11},s_1\}} & 0 \\ 0 & \begin{bmatrix} -a_{11}, e^{-s_1} & 0 & 0 \\ 0 & a_{22}, e^{s_1} & 0 \\ 0 & 0 & a_{33}, e^{s_1} \end{bmatrix} \end{bmatrix}}{\mathtt{d}\{a_{11}, s_1\}}|_{\{a_{11},s_1\}} \to 0 \\
&= \begin{bmatrix} \begin{bmatrix} 1, 1 & 0 & 0 \\ 0 & 0, -1 & 0 \\ 0 & 0 & 0, -1 \end{bmatrix} & 0 \\ 0 & \begin{bmatrix} -1, -1 & 0 & 0 \\ 0 & 0, 1 & 0 \\ 0 & 0 & 0, 1 \end{bmatrix} \end{bmatrix}.
\end{aligned}$$

If we select the sign at the exponents in different combinations, we can generate Lie operators for scaling con respect to the axes z, y, and x or also pull along an axis. Gathering the dilation Lie operators, we obtain the dilation Lie operator where the diagonal entries can be zero or ± 1

$$\mathcal{L}_{D_{a_{ii},s_i}} = \begin{bmatrix} \begin{bmatrix} 1/0, \pm 1 & 0 & 0 \\ 0 & 1/0, \pm 1 & 0 \\ 0 & 0 & 1/0, \pm 1 \end{bmatrix} & 0 \\ 0 & \begin{bmatrix} -1/0, \mp 1 & 0 & 0 \\ 0 & -1/0, \mp 1 & 0 \\ 0 & 0 & -1/0, \mp 1 \end{bmatrix} \end{bmatrix}. \tag{3.197}$$

In case of the Dilator since $e_i \wedge e_i = 0$, we will use the bivectors $e_i \wedge \bar{e}_i$ instead; thus, the table is accommodate as follows:

$$\mathcal{L}_{D_{a_{ii},s_i}} = \begin{bmatrix} & \circ & \\ \begin{bmatrix} 1/0,\pm 1 & 0 & 0 \\ 0 & 1/0,\pm 1 & 0 \\ 0 & 0 & 1/0,\pm 1 \end{bmatrix} & & \begin{bmatrix} -1/0,\mp 1 & 0 & 0 \\ 0 & -1/0,\mp 1 & 0 \\ 0 & 0 & -1/0,\mp 1 \end{bmatrix} \\ & \circ & \end{bmatrix}. \tag{3.198}$$

The versor for the dilation transformation called *Dilator* can be computed either using the bivectors $e_i\bar{e}_j$ of Eq. (3.168) or following the similar approach using the bivectors $\bar{w}_i \wedge w_j$ of Eq. (3.169) to get

$$\begin{aligned} \boldsymbol{D} &= e^{\boldsymbol{B}_D} \\ &= e^{\frac{1}{2}\{(1/0,\pm)\bar{e}_1\wedge e_1+(0/1,\mp)\bar{e}_2\wedge e_2+(0/1,\mp)\bar{e}_3\wedge e_3\}} \\ &\quad e^{\frac{1}{2}\{(-1/0,\pm)e_1\wedge\bar{e}_1+(0/-1,\mp)e_2\wedge\bar{e}_2+(0/-1,\mp)e_3\wedge\bar{e}_3\}} \\ &= e^{\frac{1}{2}\{(1/0,\pm)\bar{w}_1\wedge w_1+(0/1,\mp)\bar{w}_2\wedge w_2+(0/1,\mp)\bar{w}_3\wedge w_3\}} \\ &\quad e^{\frac{1}{2}\{(-1/0,\pm)w_1\wedge\bar{w}_1+(0/-1,\mp)w_2\wedge\bar{w}_2+(0/-1,\mp)w_3\wedge\bar{w}_3\}}. \end{aligned} \tag{3.199}$$

3.8 2D Projective Geometry in $\mathbb{R}^{3,3}$

The homography $\mathtt{H}$ in $\mathbb{P}^2$ is given by

$$\mathtt{H}_{3\times 3} = \begin{bmatrix} \mathtt{A}_{2\times 2} & \mathbf{t}_{2\times 1} \\ \mathbf{f}_{1\times 2} & h_{1\times 1} \end{bmatrix}, \tag{3.200}$$

which has 8-DOF, as an homography is a projective transformation up to a scalar factor. The projective transformation for the projective plane $\mathbb{P}^2$ can be represented using the unit basis or the null basis as follows:

$$\mathtt{T}_{e_i,\bar{e}_j} = \begin{bmatrix} a_{11} & a_{12} & 0 & 0 & 0 & f_2 \\ a_{21} & a_{22} & 0 & 0 & 0 & -f_1 \\ 0 & 0 & 0 & -f_2 & f_1 & 0 \\ 0 & 0 & -t_2 & -a_{11} & -a_{21} & 0 \\ 0 & 0 & t_1 & -a_{12} & -a_{22} & 0 \\ t_2 & -t_1 & 0 & 0 & 0 & 0 \end{bmatrix}, \tag{3.201}$$

$$\mathbb{T}_{w_i,\bar{w}_j} = \begin{bmatrix} 0 & 0 & f_2 & -a_{11} & -a_{21} & 0 \\ 0 & 0 & -f_1 & -a_{12} & -a_{22} & 0 \\ -f_2 & f_1 & 0 & 0 & 0 & 0 \\ a_{11} & a_{12} & 0 & 0 & 0 & -t_2 \\ a_{21} & a_{22} & 0 & 0 & 0 & t_1 \\ 0 & 0 & 0 & t_2 & -t_1 & 0 \end{bmatrix}. \tag{3.202}$$

Following a similar methodology for the versors obtained for the 3D projective geometry in $\mathbb{P}^3$ and using Eqs. (3.201)–(3.202), one can derive the versors for the Lie groups of $\mathbb{P}^2$, namely, for the translation (2-DOF), rotation (1-DOF) about the axis of the plane $e_1 \wedge e_2$, scaling (2-DOF), perspectivity (2-DOF), and shear (1-DOF). The derivations of these versors are left as exercises. You will see that in all these versors simply the bivectors involving the basis elements $e_3, \bar{e}_3, w_3, \bar{w}_3$ are not present. Furthermore, compare these versors with those of Sect. 3.10.5, you will see that the Lie algebra of the affine plane is exactly the same as the Lie algebra of $\mathbb{P}^2$ in $\mathbb{R}^{3,3}$.

3.9 Horosphere and *n*-Dimensional Affine Plane

This section explains briefly the meaning of the computational frameworks, the horosphere and the n-dimensional affine plane, which are useful in the study of conformal transformations [15]. This kind of transformation preserves angles between tangent vectors at each point. A common conformal transformation is the one defined by any analytic function in the complex plane. Conformal transformations also exist in the *pseudo-Euclidean* space $\mathbb{R}^{p,q}$. Since conformal transformations are nonlinear transformations, it will be desirable to linearize them. One way to do so is by moving up from the affine plane $\mathcal{A}_w(\mathbb{R}^{p,q})$ to the (p, q)-horosphere $\mathcal{H}_w^{p,q}(R^{p+1,q+1})$. Be aware that in the equations of the Sects. 3.9–3.10, we use the *Witt basis* of null vector $\{w\}$ and the corresponding reciprocal basis $\{\bar{w}\}$ of $G_{n,n}$

3.9.1 The Horosphere

The *n-dimensional affine plane* $\mathcal{A}_w(\mathbb{R}^{p,q})$ is a homogeneous representation of the points $\boldsymbol{x} \in \mathbb{R}^{p,q}$. It extends the $\mathbb{R}^{p,q}$ to a projective space with signature $\mathbb{R}^{p,q,1}$ using a null vector w as follows:

$$\boldsymbol{x}_h = \boldsymbol{x} + w \in \mathcal{A}_w(\mathbb{R}^{p,q}). \tag{3.203}$$

The $(p,\ q)$-horosphere is normally defined by

$$\mathcal{H}_w^{p,q}(R^{p+1,q+1}) = \left\{ \frac{1}{2}\boldsymbol{x}_h\overline{w}\boldsymbol{x}_h | \boldsymbol{x}_h \in \mathcal{A}_w(I\!R^{p,q}) \right\} \in R^{p+1,q+1}, \tag{3.204}$$

where the space $I\!R^{p,q}$ has been extended to $I\!R^{p+1,q+1}$ in order to have available two null vectors, $w = w_{n+1}$ and $\overline{w} = \overline{w}_{n+1}$. The conformal representation $\boldsymbol{x}_c$ of both a point $\boldsymbol{x} \in R^{p,q}$ and a point $\boldsymbol{x}_h \in \mathcal{A}_e(I\!R^{p,q})$ is given by

$$\begin{aligned} \boldsymbol{x}_c &= \frac{1}{2}\boldsymbol{x}_h\overline{w}\boldsymbol{x}_h = \frac{1}{2}[(\boldsymbol{x}_h \cdot \overline{w})\boldsymbol{x}_h + (\boldsymbol{x}_h \wedge \overline{w})\boldsymbol{x}_h] = \boldsymbol{x}_h - \frac{1}{2}\boldsymbol{x}_h^2\overline{w}, \\ &= \boldsymbol{x} - \frac{1}{2}\boldsymbol{x}_h^2\overline{w} + w = exp(\frac{1}{2}\boldsymbol{x}\overline{w})\ w\ \ exp(-\frac{1}{2}\boldsymbol{x}\overline{w}). \end{aligned} \tag{3.205}$$

This equation tells us that all points on $\mathcal{H}_w^{p,q}$ can be obtained by a simple rotation of w with respect to the plane indicated by the bivector $\boldsymbol{x}\overline{w}$.

3.9.2 The n-Dimensional Affine Plane

The points of the horosphere can be projected down into the affine plane by applying the simple formula,

$$\boldsymbol{x}_h = (\boldsymbol{x}_c \wedge \overline{w}) \cdot w, \tag{3.206}$$

and into the space $I\!R^{p,q}$ by using

$$\boldsymbol{x} = (\boldsymbol{x}_c \wedge \overline{w} \wedge w) \cdot (\overline{w} \wedge w). \tag{3.207}$$

Figure 3.1 depicts

$$\begin{aligned} \mathcal{A}_w^2 &= \{\boldsymbol{x}_h | \boldsymbol{x}_h = \boldsymbol{x} + w, \boldsymbol{x} \in I\!R^2\}, \\ \mathcal{H}_w^2 &= \{\boldsymbol{x}_c = \frac{1}{2}\boldsymbol{x}_h\overline{w}\boldsymbol{x}_h | \boldsymbol{x}_h \in \mathcal{A}_w^2\}. \end{aligned} \tag{3.208}$$

Since $\boldsymbol{x} \in I\!R^2 = span\{e_1, e_2\}$, and the null vectors computed as $w = \frac{1}{2}(e_3 + e_4)$, and $\overline{w} = e_3 - e_4$, any point on the horosphere in these terms is given by

$$\begin{aligned} \boldsymbol{x}_c &= \boldsymbol{x} - \frac{1}{2}\boldsymbol{x}^2\overline{w} + w = \boldsymbol{x} - \frac{1}{2}(\boldsymbol{x}^2 - 1)e_3 + \frac{1}{2}(\boldsymbol{x}^2 + 1)e_4, \\ &= \boldsymbol{x} + x_3e_3 + x_4e_4. \end{aligned} \tag{3.209}$$

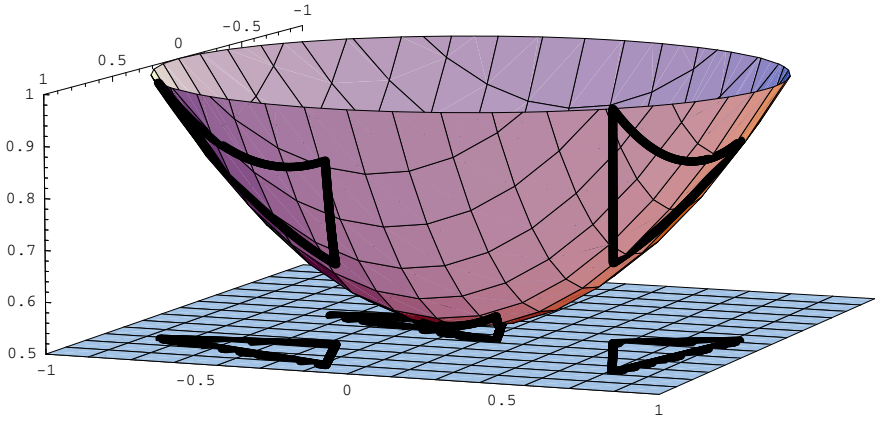


Fig. 3.1 Horosphere of R^2 with triangles of the 2D affine plane projected into the horosphere

In order to be able to depict the horosphere in 3D, in Fig. 3.1, we have ignored the coordinate x_3, considering instead the condition that e_4 is orthogonal to e_1, e_2. In Sect. 3.11, we will use the frameworks of the n-dimensional affine plane and horosphere for computations of incidence algebra.

3.10 The General Linear Algebra $gl(\mathcal{N})$ of the General Linear Lie Group $GL(\mathcal{N})$

Since each linear operator $f \in End(\mathcal{N})$ can be represented according to Eq. (3.63) in the bivector form; in this section, we study the Lie group $GL(\mathcal{N})$ and the linear Lie algebra $gl(\mathcal{N})$ using the bivector form.

3.10.1 The General Linear Lie Group $GL(\mathcal{N})$

The general linear group $GL(\mathcal{N})$ is defined to be the subset of all *endomorphisms* $f \in End(\mathcal{N})$, with the property that $f \in GL(\mathcal{N})$ if and only if $\det(f) \neq 0$ [16]. The determinant of f is defined in the algebra G_N by

$$f(w_1) \wedge f(w_2) \wedge \cdots \wedge f(w_n) = \det(\mathcal{F}) w_1 \wedge w_2 \wedge \cdots \wedge w_n, \qquad (3.210)$$

where $\det(\mathcal{F})$ is just the ordinary determinant of the matrix of f with respect to the basis $\{w\}$. Choosing the basis $\{w\}$ makes explicit the isomorphism between the general linear groups $GL(\mathcal{N})$ and $GL(n, \mathbb{C})$. The latter corresponds to the general linear group of all complex $n \times n$ matrices $\mathcal{F}$ with $\det \mathcal{F} \neq 0$. The theory of Lie

groups and their corresponding Lie algebras can be considered to be largely the study of the group manifold $GL(n, \mathbb{C})$, since any Lie group is isomorphic to a subgroup of $GL(n, \mathbb{C})$ [17, p. 501].

Since we have referred to $GL(\mathcal{N})$ as a *manifold*, we must be careful to give it the structure of an n^2-dimensional topological metric space. We define the inner product $\langle f, g\rangle$ of $f, g \in GL(\mathcal{N})$ to be the usual Hermitian positive definite inner product

$$\langle f, g\rangle = \sum_{j=1}^{n}\sum_{i=1}^{n} \overline{f_{ij}} g_{ij},$$

where $f_{ij}, g_{ij} \in \mathbb{C}$ are the components of the matrices $\mathcal{F}$ and $\mathcal{G}$ of f and g, respectively, with respect to the basis $\{w\}$. The positive definite norm $|f|$ of $f \in GL(\mathcal{N})$ is defined by

$$|f|^2 = \langle f, f\rangle = \sum_{j=1}^{n}\sum_{i=1}^{n} \overline{f_{ij}} f_{ij}$$

and is clearly zero if and only if $f = 0$.

The crucial relationship between a Lie group and its corresponding Lie algebra is almost an immediate consequence of the properties of the exponential of a linear operator $f \in End(\mathcal{N})$. The *exponential mapping* may be directly defined by the usual Taylor series

$$e^f = \sum_{i=0}^{\infty} \frac{f^i}{i!},$$

where convergence is with respect to the norm $|f|$. Note that $f^0 = 1$ is the identity operator on $\mathcal{N}$, and that f^k is the composition of f with itself k times.

The logarithm of a linear operator, $\theta_f = \log(f)$, exists and is well defined for any $f \in GL(\mathcal{N})$. The logarithm can also be defined in terms of an infinite series, or more directly, in terms of the *spectral form* of f [18]. Since the logarithm is the inverse function of the exponential function, we can write $f = e^{\theta_f}$ for any $f \in GL(\mathcal{N})$. The logarithmic form $f = e^{\theta_f}$ of $f \in GL(\mathcal{N})$ is useful for defining the *one-parameter group* $\{f_t\}$ of the operator $f \in GL(\mathcal{N})$,

$$f_t(\boldsymbol{x}) = e^{t\theta_f}\boldsymbol{x}. \tag{3.211}$$

The one-parameter group $\{f_t\}$ is *continuously connected to the identity* in the sense that $f_0(\boldsymbol{x}) = \boldsymbol{x}$, and $f_1(\boldsymbol{x}) = f(\boldsymbol{x})$. Note that

$$f_0'(\boldsymbol{x}) = \theta_f e^{t\theta_f}|_{t=0}(\boldsymbol{x}) = \theta_f(\boldsymbol{x}), \tag{3.212}$$

so θ_f is *tangent* to f_t at the identity. The reason that $\{f_t\}$ is called a one-parameter group is because it satisfies the basic additive property

$$f_s\ f_t = e^{t\theta_f} e^{s\theta_f} = e^{(s+t)\theta_f} = f_{s+t}. \tag{3.213}$$

Since each linear operator $f \in End(\mathcal{N})$ can be represented according to Eq. (3.63) in the bivector form $f(x) = F \cdot \boldsymbol{x}$, we can express the one-parameter group $g_t x = e^{tf} x$ of the skew-symmetric transformation $f(x) = F \cdot x$ in the form

$$g_t x = e^{tf} x \equiv e^{\frac{t}{2}F} x e^{-\frac{t}{2}F}. \tag{3.214}$$

This equation can be proved by showing that the terms of the Taylor series expansion of both sides of Eq. (3.214) are identical at $t = 0$.

We begin with

$$e^{tf} x \doteq e^{\frac{t}{2}F} x e^{-\frac{t}{2}F}. \tag{3.215}$$

Clearly, for $t = 0$, we have

$$e^{0f} x = e^{0F} x e^{0F} = x.$$

Next, taking the first derivative of both sides of Eq. (3.215), we get

$$e^{tf} fx \doteq \frac{1}{2} F e^{\frac{t}{2}F} x e^{-\frac{t}{2}F} - \frac{1}{2} F e^{\frac{t}{2}F} x e^{-\frac{t}{2}F} = e^{\frac{t}{2}F} (F \cdot x) e^{-\frac{t}{2}F}. \tag{3.216}$$

Setting t equal to zero gives the identity $f(x) = F \cdot x$.

Taking the derivative of both sides of (3.216) gives

$$e^{tf} f^2 x \doteq \frac{1}{2} F e^{\frac{t}{2}F} (F \cdot x) e^{-\frac{t}{2}F} - \frac{1}{2} F e^{\frac{t}{2}F} (F \cdot x) e^{-\frac{t}{2}F} = e^{\frac{t}{2}F} [F \cdot (F \cdot x)] e^{-\frac{t}{2}F},$$

and setting t equal to zero gives the identity $f^2(x) = F \cdot (F \cdot x)$. Continuing to take successive derivatives of (3.215) gives

$$e^{tf} f^k(x) \doteq e^{\frac{t}{2}F} (F^k : x) e^{-\frac{t}{2}F}, \tag{3.217}$$

where $F^k : x$ is defined recursively by $F^1 : x = F \cdot x$ and

$$F^k : x = F \cdot (F^{k-1} : x). \tag{3.218}$$

Finally, setting t equal to zero in Eq. (3.217) gives the identity

$$f^k(x) = F^k : x.$$

Equation (3.218) is interesting because it expresses the powers of a linear operator in terms of "powers" of its defining bivector. It is clear that each bivector defines a unique skew-symmetric linear operator, and conversely, that each skew-symmetric linear operator defines a unique bivector (see Eq. (3.63)). Thus, the study of the

structure of a bivector is determined by and uniquely determines the corresponding structure of the corresponding linear operator. The proof of the above theorem is attributable to Marcel Riesz [19].

3.10.2 The General Linear Algebra $gl(\mathcal{N})$

We can now define the *general linear Lie algebra* $gl(\mathcal{N})$ of the general linear Lie group $GL(\mathcal{N})$. As a set, $gl(\mathcal{N}) \equiv End(\mathcal{N})$, which is just the set of all *tangent operators* $\theta_f = \log(f) \in End(\mathcal{N})$ to the one-parameter groups $f_t = e^{t\theta_f}$ defined for each $f \in GL(\mathcal{N})$. But to complete the definition of $gl(\mathcal{N})$, we must specify the algebraic operations of addition and multiplication which allow $End(\mathcal{N})$ to be seen as the Lie algebra $gl(\mathcal{N})$. Addition requires only the ordinary addition of linear operators, but multiplication is defined by the *Lie bracket* $[\theta_f, \theta_g]$ for $\theta_f, \theta_g \in gl(\mathcal{N})$. We will give an analytic definition of the Lie bracket [20, p. 3] which directly ties it to the group structure of $GL(\mathcal{N})$

$$[\theta_f, \theta_g] = \frac{d}{d(t^2)} f_t g_t f_{-t} g_{-t}|_{t=0} = \frac{1}{2t}\frac{d}{dt} f_t g_t f_{-t} g_{-t}|_{t=0}.$$

Evaluating the Lie bracket by using the Taylor series expansions,

$$\begin{aligned}
[\theta_f, \theta_g] &= \frac{1}{2t}\frac{d}{dt}(f_t g_t f_{-t} g_{-t})\big|_{t=0}, \\
&= \frac{1}{2t}\big(\theta_f f_t g_t f_{-t} g_{-t} + f_t \theta_g g_t f_{-t} g_{-t} - f_t g_t \theta_f f_{-t} g_{-t} - f_t g_t f_{-t} \theta_g g_{-t}\big), \\
&= \left(\frac{1}{2t} f_t(\theta_f g_t - g_t \theta_f) f_{-t} g_{-t}\right)|_{t=0} + \left(\frac{1}{2t} f_t g_t(\theta_g f_{-t} - f_{-t}\theta_g) g_{-t}\right)|_{t=0}, \\
&= \frac{1}{2}(\theta_f \theta_g - \theta_g \theta_f) + \frac{1}{2}(-\theta_g \theta_f + \theta_f \theta_g), \\
&= \theta_f \theta_g - \theta_g \theta_f, \qquad (3.219)
\end{aligned}$$

we find that

$$g_t = 1 + t\theta_g + \cdots, \quad \text{and} \quad f_{-t} = 1 - t\theta_f + \cdots.$$

We have thus demonstrated that the Lie bracket, defined analytically above, reduces simply to the commutator product of the linear operators θ_f and θ_g in $gl(\mathcal{N})$. As such, it is not difficult to show that it satisfies the Jacoby identity of Eq. (3.11), which is equivalent to the distributive law

$$[\theta_f, [\theta_g, \theta_h]] = [[\theta_f, \theta_g], \theta_h] + [\theta_g, [\theta_f, \theta_h]]. \qquad (3.220)$$

When we choose a particular basis $\{w\}$ of $\mathcal{N}$, the isomorphism between the general linear Lie algebra $gl(n, \mathbb{C})$ and $gl(\mathcal{N})$ becomes explicit, and the Lie bracket of linear operators just becomes the Lie bracket of $n \times n$ matrices,

$$[f, g]\{w\} = fg\{w\} - gf\{w\} = \{w\}(\mathcal{FG} - \mathcal{GF}) = \{w\}[\mathcal{F}, \mathcal{G}], \tag{3.221}$$

where $[\mathcal{F}, \mathcal{G}]$ are the commutator products of the matrices $\mathcal{F}$ and $\mathcal{G}$. Alternatively, using the bivector representation of Eq. (3.63), the Lie bracket of linear operators is expressed in terms of the Lie bracket of the bivectors of the operators (3.67).

3.10.3 The Orthogonal Groups

The simplest well-known example of an *orthogonal group* is $SO(2)$, which is a subgroup of the general linear group $GL(\mathcal{N}^2)$. As a matrix group, it is generated by all 2×2 matrices of the form

$$X_\theta = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}. \tag{3.222}$$

The matrix X_θ generates a counterclockwise rotation in the xy-plane through the angle θ. Using (3.63), we get the corresponding bivector representation

$$\boldsymbol{X}_\theta = \cos(\theta) w_1 \wedge \bar{w}_1 - \sin(\theta) w_1 \wedge \bar{w}_2 + \sin(\theta) w_2 \wedge \bar{w}_1 + \cos(\theta) w_2 \wedge \bar{w}_2. \tag{3.223}$$

For matrices $X_{\theta_1}, X_{\theta_2} \in SO(2)$, the group operation is ordinary matrix multiplication, $X_{\theta_2} X_{\theta_1} = X_{\theta_1 + \theta_2}$. For the bivector representation $\boldsymbol{X}_{\theta_1}, \boldsymbol{X}_{\theta_2} \in SO(2)$, the group operation is defined by its *generalized dot product*, that is, for $\boldsymbol{x} \in \mathcal{N}^2$,

$$(\boldsymbol{X}_{\theta_1} : \boldsymbol{X}_{\theta_2}) \equiv \boldsymbol{X}_{\theta_2} \cdot (\boldsymbol{X}_{\theta_1} \cdot \boldsymbol{x}) = \boldsymbol{X}_{\theta_1 + \theta_2} \cdot \boldsymbol{x}. \tag{3.224}$$

Note that the bivectors $\boldsymbol{X}_\theta$ are in $G^2_{n,n}$.

Taking the derivatives of X_θ and $\boldsymbol{X}_\theta$ with respect to θ and evaluating at $\theta = 0$ gives the corresponding generators of the associated Lie algebra $so(2)$. As a matrix Lie algebra under the bracket operation of matrices, we find the generator

$$\frac{dX_\theta}{d\theta}|_{\theta \to 0} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}. \tag{3.225}$$

As a bivector Lie algebra under the bracket operation of bivectors, we use Eq. (3.223) to find the bivector generator

$$\boldsymbol{B} = \frac{d\boldsymbol{X}_\theta}{d\theta}|_{\theta \to 0} = -w_1 \wedge \bar{w}_2 + w_2 \wedge \bar{w}_1 = -e_1 e_2 + \bar{e}_1 \bar{e}_2. \tag{3.226}$$

The *spinor group* $Spin(2)$ is defined by taking the exponential of the bivector (see Eq. (3.226)),

$$Spin(2) = \{\exp(\frac{1}{2}\theta \boldsymbol{B}) |\ \ \theta \in I\!R\}.$$

According to Sobczyk [21], the exponential $\exp(\frac{1}{2}\theta \boldsymbol{B})$ can be calculated by noting that the bivector $\boldsymbol{B}$ satisfies the *minimal polynomial*

$$\boldsymbol{B}^3 + 4\boldsymbol{B} = \boldsymbol{B}(\boldsymbol{B} - 2i)(\boldsymbol{B} + 2i) = 0,$$

which implies the decomposition

$$\boldsymbol{B} = 0p_1 + 2ip_2 - 2ip_3,$$

where the mutually annihilating idempotents are defined by

$$p_1 = \frac{\boldsymbol{B}^2 + 4}{4}, \quad p_2 = -\frac{1}{8}\boldsymbol{B}(\boldsymbol{B} + 2i), \quad p_3 = -\frac{1}{8}\boldsymbol{B}(\boldsymbol{B} - 2i).$$

Using this decomposition, we find that

$$\begin{aligned} \exp\left(\frac{1}{2}\theta \boldsymbol{B}\right) &= \exp\left(\frac{0 \cdot \theta}{2}\right) p_1 + \exp(i\theta)p_2 + \exp(-i\theta)p_3, \\ &= p_1 + \cos(\theta)(p_2 + p_3) + \sin(\theta)i(p_2 - p_3), \\ &= p_1 + \cos(\theta)(p_2 + p_3) + \frac{\boldsymbol{B}}{2}\sin(\theta). \end{aligned} \tag{3.227}$$

The group action of the *spinor group* $Spin(2)$ is given by

$$\boldsymbol{x}' = \exp\left(\frac{1}{2}\theta \boldsymbol{B}\right) \boldsymbol{x} \exp\left(-\frac{1}{2}\theta \boldsymbol{B}\right),$$

where $\boldsymbol{x} = \{w\}x_{\{w\}} = x_1 w_1 + x_2 w_2$. We say that $Spin(2)$ is a "double covering" of the orthogonal group $SO(2)$ because the spinors $\pm\exp\left(\frac{1}{2}\theta \boldsymbol{B}\right)$ represent the same group element. Note that now we can formulate the easy rule for the composition of two group elements, $\exp\left(\frac{1}{2}\theta_1 \boldsymbol{B}\right)$ and $\exp\left(\frac{1}{2}\theta_2 \boldsymbol{B}\right)$,

$$\exp\left(\frac{1}{2}\theta_1 \boldsymbol{B}\right) \exp\left(\frac{1}{2}\theta_2 \boldsymbol{B}\right) = \exp\left(\frac{1}{2}(\theta_1 + \theta_2)\boldsymbol{B}\right).$$

If we are only interested in the group $SO(2)$, a more natural place to carry out the calculations is in the Euclidean space $I\!R^2$. We project the null cone $\mathcal{N}^2$ down to $I\!R^2$ by using the reciprocal pseudoscalars I_2 and $\overline{I_2}$ defined by

$$I_2 = e_1 e_2 \quad \text{and} \quad \overline{I_2} = (2 - \sqrt{2})^2(\bar{w}_2 + e_2)(\bar{w}_1 + e_1).$$

Thus, for $\boldsymbol{x} = \{w\}x_{\{w\}} = x_1 w_1 + x_2 w_2 \in \mathcal{N}^2$, the projection $\boldsymbol{x}' = P_I(\boldsymbol{x})$ gives

$$\boldsymbol{x}' = P_I(\boldsymbol{x}) = (\boldsymbol{x} \cdot \overline{I}) \cdot I = x_1 e_1 + x_2 e_2 \in I\!R^2.$$

Note that this projection is invertible, in the sense that we can find $P_{I'}$ such that $\boldsymbol{x} = P_{I'}(\boldsymbol{x}')$. The projection $P_{I'}$ is specified by

$$\boldsymbol{x} = P_{I'}(\boldsymbol{x}') = (\boldsymbol{x}' \cdot \overline{I}) \cdot I' = x_1 w_1 + x_2 w_2, \tag{3.228}$$

where $\overline{I}$ is defined as before and where $I' = w_1 w_2$.

In $I\!R^2$, the generator of rotations is the simple bivector $e_2 e_1$. This bivector can be obtained from the bivector (3.226) in $spin(2, 2)$ by the simple projection

$$P_I(\boldsymbol{B}) = I_2^{-1} I_2 \cdot \boldsymbol{B} = e_2 e_1 = -I_2 \tag{3.229}$$

onto the Lie algebra $so(2)$. For $\boldsymbol{x}' = x_1 e_1 + x_2 e_2 \in I\!R^2$, the equivalent rotation is given by

$$\boldsymbol{y}' = \exp\left(\frac{1}{2}\theta e_2 e_1\right) \boldsymbol{x}' \exp\left(-\frac{1}{2}\theta e_2 e_1\right). \tag{3.230}$$

The above ideas can be immediately generalized to the general Lie group $GL(\mathcal{N}^n)$ of null cone $\mathcal{N}^n$ and the orthogonal subgroups $SO(p, q)$ where $p + q = n$. The orthogonal group $SO(p, q)$ acts on the space $I\!R^{p,q}$. Thus, if we wish to work in this Lie group or in the corresponding Lie algebra, we first project the null cone $\mathcal{N}^n$ onto $I\!R^{p,q}$ by using the reciprocal vector basis elements, then carry out the rotation, and finally return to the null cone by using the inverse projection.

3.10.4 Computing Rigid Motion in the Affine Plane

A rotation in the affine n-plane $\mathcal{A}_w^n = \mathcal{A}_w(I\!R^n)$, just as in the Euclidean space $I\!R^n$, is the product of two reflections through two intersecting hyperplanes. If the normal unit vectors to these hyperplanes are m and n, respectively, then the versor of the rotation is given by

$$R = mn = e^{\frac{\theta}{2}B} = \cos\left(\frac{\theta}{2}\right) + B \sin\left(\frac{\theta}{2}\right), \tag{3.231}$$

where B is the unit bivector defining the plane of the rotation.

A translation of the vector $x_h \in \mathcal{A}_w^n$, along the vector $t \in I\!R^n$, to the vector $x_h' = x_h + t \in \mathcal{A}_w^n$, is affected by the versor

$$T = \exp\left(\frac{1}{2}t\bar{w}\right) = 1 + \frac{1}{2}t\bar{w}, \tag{3.232}$$

when it is followed by the projection $P_A(x') \equiv (x \wedge \bar{w}) \cdot w$, which brings the horosphere *back* into the affine plane. Thus, for $x_h \in \mathcal{A}^n_w$, we get

$$\begin{aligned} x' &= TxT^{-1} = \exp\left(\frac{1}{2}t\bar{w}\right) x_h \exp\left(-\frac{1}{2}t\bar{w}\right), \\ &= \left(1 + \frac{1}{2}t\bar{w}\right) x_h \left(1 - \frac{1}{2}t\bar{w}\right) = x_h + \frac{1}{2}t\bar{w}x_h - \frac{1}{2}x_h t\bar{w} - \frac{1}{4}t\bar{w}x_h t\bar{w}, \\ &= x_h + t + t \cdot (\bar{w} \wedge x^h) - \frac{1}{2}t^2\bar{w}, \\ &= x_h + t - \left(t \cdot x_h + \frac{1}{2}t^2\right)\bar{w}. \end{aligned} \tag{3.233}$$

Applying P_A to this result, we get the expected translated vector

$$x'_h = P_A(x') = P_A\left[x_h + t - \left(t \cdot x_h + \frac{1}{2}t^2\right)\bar{w}\right] = x_h + t. \tag{3.234}$$

The advantage of carrying out translations in the affine plane rather than in the horosphere is that the affine plane is basically still a linear model of Euclidean space, whereas the horosphere is a more complicated *nonlinear* model.

Combining the versors for a rotation and a translation, we get the expression for the versor $M = TR$ of a rigid motion. For $x_h \in \mathcal{A}^n_w$, we then find that

$$x'_h = P_A[Mx_hM^{-1}] = P_A[TRx_hR^{-1}T^{-1}]. \tag{3.235}$$

Equivalently, we will often write $M^{-1} \equiv \widetilde{M}$, expressing M^{-1} in terms of the operation of *conjugation*. Whenever a calculation involves a translation, we must always apply the projection P_A to guarantee that our end result will be in the affine plane. The above calculations can be checked with the Clifford algebra calculator CLICAL 4.0 [22]. Comparisons can also be made to the corresponding calculations made by Hestenes and Li [23] on the horosphere.

3.10.5 The Lie Algebra of the Affine Plane

The Lie algebra of the neutral affine plane $\mathcal{A}_{w_3}(\mathcal{N}^2)$ is useful in the analysis of visual invariants [24], so we will begin with its treatment here. An homography $\mathtt{H}$ in $\mathbb{P}^2$ is given by

$$\mathtt{H} = \begin{bmatrix} \mathtt{A} & \mathbf{t} \\ \mathbf{f} & h \end{bmatrix}, \tag{3.236}$$

where $\mathtt{A}$ is an affine transformation matrix, and $\mathbf{t}$ is a 2D translation vector, and if $h = 1$ the vector $\mathbf{f} = \mathtt{0}$ corresponds to the well-known matrix representation of the Lie group of affine transformations in the plane that has six independent parameters, or degrees of freedom, and consists of all matrices of the form

$$g(A, \boldsymbol{v}) = \begin{bmatrix} a_{11} & a_{12} & a \\ a_{21} & a_{22} & b \\ 0 & 0 & 1 \end{bmatrix}, \tag{3.237}$$

where $\det g(A, \boldsymbol{v}) = \det A \neq 0$.

The one-parameter subgroups are generated by the matrices

$$\begin{aligned} T_x &= \begin{bmatrix} 1 & 0 & x \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, & T_y &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & y \\ 0 & 0 & 1 \end{bmatrix}, \\ D_u &= \begin{bmatrix} e^u & 0 & 0 \\ 0 & e^u & 0 \\ 0 & 0 & 1 \end{bmatrix}, & R_\theta &= \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}, \\ S_v &= \begin{bmatrix} e^v & 0 & 0 \\ 0 & e^{-v} & 0 \\ 0 & 0 & 1 \end{bmatrix}, & H_\phi &= \begin{bmatrix} \cosh(\phi) & \sinh(\phi) & 0 \\ \sinh(\phi) & \cosh(\phi) & 0 \\ 0 & 0 & 1 \end{bmatrix}. \end{aligned} \tag{3.238}$$

Using Eq. (3.212), we obtain the matrix representation of the Lie algebra basis generators by taking the derivative of Eq. (3.238) and evaluating the parameter at zero:

$$\begin{aligned} \mathcal{L}_x &= \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, & \mathcal{L}_y &= \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}, \\ \mathcal{L}_s &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}, & \mathcal{L}_r &= \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \\ \mathcal{L}_b &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix}, & \mathcal{L}_B &= \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}. \end{aligned} \tag{3.239}$$

The above matrix Lie group and matrix Lie algebra can be directly translated into the corresponding Lie group and Lie algebra of the affine plane $\mathcal{A}_{w_3}(\mathcal{N}^2)$. Each of the matrix generators in (3.238) and (3.239) can be replaced by its corresponding bivector

representation (3.63). For example, the bivector representations of the generators of the Lie algebra are

$$\begin{aligned}
\mathcal{L}_x &= bivector(\mathcal{L}_x) = w_1 \wedge \bar{w}_3,\\
\mathcal{L}_y &= bivector(\mathcal{L}_y) = w_2 \wedge \bar{w}_3,\\
\mathcal{L}_s &= bivector(\mathcal{L}_s) = w_1 \wedge \bar{w}_1 + w_2 \wedge \bar{w}_2,\\
\mathcal{L}_r &= bivector(\mathcal{L}_r) = w_2 \wedge \bar{w}_1 - w_1 \wedge \bar{w}_2,\\
\mathcal{L}_b &= bivector(\mathcal{L}_b) = w_1 \wedge \bar{w}_1 - w_2 \wedge \bar{w}_2,\\
\mathcal{L}_B &= bivector(\mathcal{L}_\phi) = w_1 \wedge \bar{w}_2 + w_2 \wedge \bar{w}_1.
\end{aligned} \tag{3.240}$$

Expanding these bivector generators in the standard basis (3.68), we get

$$\begin{aligned}
\mathcal{L}_x &= \tfrac{1}{2}e_1e_3 - \tfrac{1}{2}e_1\bar{e}_3 - \tfrac{1}{2}e_3\bar{e}_1 - \tfrac{1}{2}\bar{e}_1\bar{e}_3,\\
\mathcal{L}_y &= \tfrac{1}{2}e_2e_3 - \tfrac{1}{2}e_2\bar{e}_3 - \tfrac{1}{2}e_3\bar{e}_2 - \tfrac{1}{2}\bar{e}_2\bar{e}_3,\\
\mathcal{L}_s &= -e_1\bar{e}_1 - e_2\bar{e}_2,\\
\mathcal{L}_r &= -e_1e_2 + \bar{e}_1\bar{e}_2,\\
\mathcal{L}_b &= -e_1\bar{e}_1 + e_2\bar{e}_2,\\
\mathcal{L}_B &= -e_1\bar{e}_2 - e_2\bar{e}_1.
\end{aligned} \tag{3.241}$$

Let us see how the Lie algebra of the affine plane can be represented as a Lie algebra of vector fields over the null cone $\mathcal{N}^3$. The *vector derivative* or *gradient* $\partial_{\boldsymbol{x}} = \frac{\partial}{\partial \boldsymbol{x}}$ at the point $\boldsymbol{x} = xw_1 + yw_2 + zw_3 \in \mathcal{N}^3$ is defined by requiring $\boldsymbol{a} \cdot \partial_{\boldsymbol{x}}$ to be the *directional derivative* in the direction of $\boldsymbol{a}$. It follows that $\boldsymbol{a} \cdot \partial_{\boldsymbol{x}} \boldsymbol{x} = \boldsymbol{a}$. We also have

$$\partial_{\boldsymbol{x}} \boldsymbol{x} = \partial_{\boldsymbol{x}} \cdot \boldsymbol{x} + \partial_{\boldsymbol{x}} \wedge \boldsymbol{x} = 3 + \sum_{i=1}^{3} \bar{w}_i \wedge\ w_i,$$

where $\{w\}$ and $\{\bar{w}\}$ are reciprocal bases for the reciprocal null cones $\mathcal{N}^3$ and $\overline{\mathcal{N}^3}$.

Now, let $\boldsymbol{a} = \boldsymbol{a}(\boldsymbol{x})$ and $\boldsymbol{b} = \boldsymbol{b}(\boldsymbol{x})$ be vector fields in $\mathcal{N}^3$. The Lie bracket $[\boldsymbol{a}, \boldsymbol{b}]$ is defined by

$$[\boldsymbol{a}, \boldsymbol{b}] = \boldsymbol{a} \cdot \partial_{\boldsymbol{x}} \boldsymbol{b} - \boldsymbol{b} \cdot \partial_{\boldsymbol{x}} \boldsymbol{a}.$$

Since in $\mathcal{N}^3$, $\partial_{\boldsymbol{x}} \wedge \partial_{\boldsymbol{x}} = 0$, we have the important integrability condition that

$$(\boldsymbol{a} \wedge \boldsymbol{b}) \cdot (\partial_{\boldsymbol{x}} \wedge \partial_{\boldsymbol{x}}) = [\boldsymbol{a}, \boldsymbol{b}] \cdot \partial_{\boldsymbol{x}} - [\boldsymbol{a} \cdot \partial_{\boldsymbol{x}}, \boldsymbol{b} \cdot \partial_{\boldsymbol{x}}] = 0,$$

where

$$[\boldsymbol{a} \cdot \partial_{\boldsymbol{x}}, \boldsymbol{b} \cdot \partial_{\boldsymbol{x}}] = \boldsymbol{a} \cdot \partial_{\boldsymbol{x}} \boldsymbol{b} \cdot \partial_{\boldsymbol{x}} - \boldsymbol{b} \cdot \partial_{\boldsymbol{x}} \boldsymbol{a} \cdot \partial_{\boldsymbol{x}}$$

is the Lie bracket or commutator product of the partial derivatives $\boldsymbol{a} \cdot \partial_{\boldsymbol{x}}$ and $\boldsymbol{b} \cdot \partial_{\boldsymbol{x}}$. It follows from this identity that

$$[\boldsymbol{a}, \boldsymbol{b}] \cdot \partial_{\boldsymbol{x}} = [\boldsymbol{a} \cdot \partial_{\boldsymbol{x}}, \boldsymbol{b} \cdot \partial_{\boldsymbol{x}}],$$

which relates the Lie bracket of the vector fields $[\boldsymbol{a}, \boldsymbol{b}]$ to the standard Lie bracket of the partial derivatives $[\boldsymbol{a} \cdot \partial_{\boldsymbol{x}}, \boldsymbol{b} \cdot \partial_{\boldsymbol{x}}]$.

Let us consider in detail the translation of the Lie algebra of the affine plane to the null vector formulation in the null cone $\mathcal{N}^2$. Recall that the two-dimensional affine plane $\mathcal{A}_e(\mathcal{N}^2)$ in $\mathcal{N}^3$ is defined by

$$\mathcal{A}_e(\mathcal{N}) = \{\boldsymbol{x} \in \mathcal{N}^3 |\ \boldsymbol{x} = xw_1 + yw_2 + w_3\}. \tag{3.242}$$

We have already seen that the Lie algebra of the affine plane can be defined by a Lie algebra of matrices, or by an equivalent Lie algebra of bivectors. We now define this same Lie algebra as a Lie algebra of partial derivatives or as a Lie algebra of vector fields. We have the following correspondences:

$$\mathcal{L}_x = \frac{\partial}{\partial x} = w_1 \cdot \partial_{\boldsymbol{x}} = \boldsymbol{\mathcal{L}}_x \cdot (\boldsymbol{x} \wedge \partial_{\boldsymbol{x}}) \leftrightarrow \mathcal{L}_x \boldsymbol{x} = \boldsymbol{\mathcal{L}}_x \cdot \boldsymbol{x} = w_1 = L_x, \tag{3.243}$$

where $\boldsymbol{\mathcal{L}}_x = w_1 \wedge \bar{w}_3$;

$$\mathcal{L}_y = \frac{\partial}{\partial y} = w_2 \cdot \partial_{\boldsymbol{x}} = \boldsymbol{\mathcal{L}}_y \cdot (\boldsymbol{x} \wedge \partial_{\boldsymbol{x}}) \leftrightarrow \mathcal{L}_y \boldsymbol{x} = \boldsymbol{\mathcal{L}}_y \cdot \boldsymbol{x} = w_2 = L_y, \tag{3.244}$$

where $\boldsymbol{\mathcal{L}}_y = w_2 \wedge \bar{w}_3$;

$$\begin{aligned} \mathcal{L}_s &= x\frac{\partial}{\partial x} + y\frac{\partial}{\partial y} = (\boldsymbol{x} - w_3) \cdot \partial_{\boldsymbol{x}} = \boldsymbol{\mathcal{L}}_s \cdot (\boldsymbol{x} \wedge \partial_{\boldsymbol{x}}) \\ &\leftrightarrow \mathcal{L}_s \boldsymbol{x} = \boldsymbol{\mathcal{L}}_s \cdot \boldsymbol{x} = xw_1 + yw_2 = \boldsymbol{x} - w_3 = L_s, \end{aligned} \tag{3.245}$$

where $\boldsymbol{\mathcal{L}}_s = w_1 \wedge \bar{w}_1 + w_2 \wedge \bar{w}_2$;

$$\mathcal{L}_r = -y\frac{\partial}{\partial x} + x\frac{\partial}{\partial y} = \boldsymbol{\mathcal{L}}_r \cdot (\boldsymbol{x} \wedge \partial_{\boldsymbol{x}}) \leftrightarrow \mathcal{L}_r \boldsymbol{x} = \boldsymbol{\mathcal{L}}_r \cdot \boldsymbol{x} = L_r, \tag{3.246}$$

where $\boldsymbol{\mathcal{L}}_r = w_2 \wedge \bar{w}_1 - w_1 \wedge \bar{w}_2$;

$$\mathcal{L}_b = x\frac{\partial}{\partial x} - y\frac{\partial}{\partial y} = \boldsymbol{\mathcal{L}}_b \cdot (\boldsymbol{x} \wedge \partial_{\boldsymbol{x}}) \leftrightarrow \mathcal{L}_b \boldsymbol{x} = \boldsymbol{\mathcal{L}}_b \cdot \boldsymbol{x} = L_b, \tag{3.247}$$

where $\boldsymbol{\mathcal{L}}_b = w_1 \wedge \bar{w}_1 - w_2 \wedge \bar{w}_2$; and

$$\mathcal{L}_B = y\frac{\partial}{\partial x} + x\frac{\partial}{\partial y} = \boldsymbol{\mathcal{L}}_B \cdot (\boldsymbol{x} \wedge \partial_{\boldsymbol{x}}) \leftrightarrow \mathcal{L}_B \boldsymbol{x} = \boldsymbol{\mathcal{L}}_B \cdot \boldsymbol{x} = L_B, \tag{3.248}$$

where $\boldsymbol{\mathcal{L}}_B = w_1 \wedge \bar{w}_2 + w_2 \wedge \bar{w}_1$.

Thus, the Lie algebra of the affine plane is generated by the bivectors

$$\mathcal{M}_{bivectors} = \{\mathcal{L}_x, \mathcal{L}_y, \mathcal{L}_s, \mathcal{L}_r, \mathcal{L}_b, \mathcal{L}_B\}, \tag{3.249}$$

or, equivalently, by the vector fields of the form

$$\begin{aligned} \mathcal{M}_{vectorfields} &= \{\mathcal{L}_x \cdot \boldsymbol{x}, \mathcal{L}_y \cdot \boldsymbol{x}, \mathcal{L}_s \cdot \boldsymbol{x}, \mathcal{L}_r \cdot \boldsymbol{x}, \mathcal{L}_b \cdot \boldsymbol{x}, \mathcal{L}_B \cdot \boldsymbol{x}\}, \\ &= \{L_x, L_y, L_s, L_r, L_b, L_B\}, \end{aligned} \tag{3.250}$$

where $\mathcal{L} \cdot \boldsymbol{x}$ for $\mathcal{L} \in \mathcal{M}_{bivectors}$.

The Lie bracket $[\mathcal{L}_1 \cdot \boldsymbol{x}, \mathcal{L}_2 \cdot \boldsymbol{x}]$ is given by

$$[\mathcal{L}_1 \cdot \boldsymbol{x}, \mathcal{L}_2 \cdot \boldsymbol{x}] = \mathcal{L}_2 \cdot (\mathcal{L}_1 \cdot \boldsymbol{x}) - \mathcal{L}_1 \cdot (\mathcal{L}_2 \cdot \boldsymbol{x}) = (\mathcal{L}_2 \times \mathcal{L}_1) \cdot \boldsymbol{x},$$

where $\mathcal{L}_1 \times \mathcal{L}_2 = \frac{1}{2}(\mathcal{L}_1\mathcal{L}_2 - \mathcal{L}_2\mathcal{L}_1)$ is the commutator product of the bivectors $\mathcal{L}_1, \mathcal{L}_2 \in \mathcal{M}$.

The Lie algebra of the affine plane is useful for the analysis of motion in the image plane [24]. The vector fields of this Lie algebra are tangent to the flows or integral curves of their group action on the manifold and are presented in Fig. 3.2 as real images.

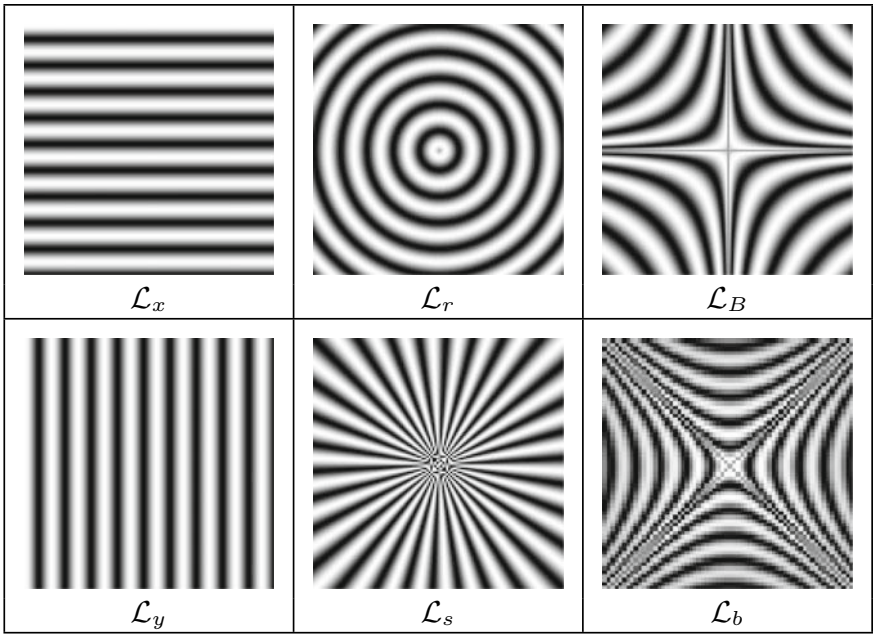


Fig. 3.2 Lie algebra basis in the form of real images

Table 3.1 Lie algebra of the affine plane

$[\cdot,\cdot]$	$\mathcal{L}_x$	$\mathcal{L}_y$	$\mathcal{L}_s$	$\mathcal{L}_r$	$\mathcal{L}_b$	$\mathcal{L}_B$
$\mathcal{L}_x$	0	0	$\mathcal{L}_x$	$\mathcal{L}_y$	$\mathcal{L}_x$	$\mathcal{L}_y$
$\mathcal{L}_y$	0	0	$\mathcal{L}_y$	$-\mathcal{L}_x$	$-\mathcal{L}_y$	$\mathcal{L}_x$
$\mathcal{L}_s$	$-\mathcal{L}_x$	$-\mathcal{L}_y$	0	0	0	0
$\mathcal{L}_r$	$-\mathcal{L}_y$	$\mathcal{L}_x$	0	0	$-2\mathcal{L}_B$	$2\mathcal{L}_b$
$\mathcal{L}_b$	$-\mathcal{L}_x$	$\mathcal{L}_y$	0	$2\mathcal{L}_B$	0	$2\mathcal{L}_r$
$\mathcal{L}_B$	$-\mathcal{L}_y$	$-\mathcal{L}_x$	0	$-2\mathcal{L}_b$	$-2\mathcal{L}_r$	0

We have found the generators

$$\begin{array}{|l|l|l|}\hline \mathcal{L}_x = \frac{\partial}{\partial x} & \mathcal{L}_r = -y\frac{\partial}{\partial x} + x\frac{\partial}{\partial y} & \mathcal{L}_B = x\frac{\partial}{\partial x} - y\frac{\partial}{\partial y} \\ \hline \mathcal{L}_y = \frac{\partial}{\partial y} & \mathcal{L}_s = x\frac{\partial}{\partial x} + y\frac{\partial}{\partial y} & \mathcal{L}_b = y\frac{\partial}{\partial x} + x\frac{\partial}{\partial y} \\ \hline \end{array} \tag{3.251}$$

of the Lie algebra of the affine plane $\mathcal{A}_{w_3}(\mathcal{N}^2)$ as vector fields along integral curves. Taking the commutator products of these infinitesimal differential generators gives the following multiplication Table 3.1 for this Lie algebra.

Using Table 3.1, we can verify the Jacobi identity for $\mathcal{L}_x$, $\mathcal{L}_s$, and $\mathcal{L}_b$, getting

$$\begin{array}{ccccccc} [\mathcal{L}_x[\mathcal{L}_s\mathcal{L}_b]] & + & [\mathcal{L}_s[\mathcal{L}_b\mathcal{L}_x]] & + & [\mathcal{L}_b[\mathcal{L}_x\mathcal{L}_s]] & = & \\ [\mathcal{L}_x 0] & - & [\mathcal{L}_s\mathcal{L}_y] & + & [\mathcal{L}_b\mathcal{L}_x] & = & \\ 0 & + & \mathcal{L}_y & - & \mathcal{L}_y & = & 0. \end{array} \tag{3.252}$$

Or, equivalently, using CLICAL and the bivector representation for $\boldsymbol{\mathcal{L}}_x$, $\boldsymbol{\mathcal{L}}_r$, and $\boldsymbol{\mathcal{L}}_b$, we calculate

$$\begin{array}{ccccccc} [\boldsymbol{\mathcal{L}}_x[\boldsymbol{\mathcal{L}}_r\boldsymbol{\mathcal{L}}_b]] & + & [\boldsymbol{\mathcal{L}}_r[\boldsymbol{\mathcal{L}}_b\boldsymbol{\mathcal{L}}_x]] & + & [\boldsymbol{\mathcal{L}}_b[\boldsymbol{\mathcal{L}}_x\boldsymbol{\mathcal{L}}_r]] & = & \\ 2[\boldsymbol{\mathcal{L}}_x\boldsymbol{\mathcal{L}}_B] & - & [\boldsymbol{\mathcal{L}}_r\boldsymbol{\mathcal{L}}_y] & + & [\boldsymbol{\mathcal{L}}_b\boldsymbol{\mathcal{L}}_y] & = & \\ 2\boldsymbol{\mathcal{L}}_x & - & \boldsymbol{\mathcal{L}}_x & - & \boldsymbol{\mathcal{L}}_x & = & 0. \end{array} \tag{3.253}$$

3.11 The Algebra of Incidence

In various applications in robotics, image analysis, and computer vision, the use of projective geometry and the algebra of incidence is extreme useful. Fortunately, these mathematical systems can be efficiently handled within the geometric algebra framework.

In projective geometry, points are represented using homogeneous coordinates of nonzero vectors in the $(n+1)$-dimensional Euclidean space $\mathbb{R}^{n+1}$. These can be seen as *projective rays* identified as points in the n-dimensional projective plane

Π^n of $\mathbb{R}^{n+1}$. Furthermore, points, lines, planes, and higher dimensional k-planes in Π^n are related with 1, 2, 3, and $(k+1)$-dimensional subspaces S^r of $\mathbb{R}^{n+1}$, where $k \leq n$. Since each k-subspace can be associated with a nonzero k-blade A_k of the geometric algebra $G(\mathbb{R}^{n+1})$, it follows that the corresponding $(k-1)$-plane in Π^n can be named by the k-direction of the k-blade A_k.

The *meet* and *join* in Π^n are the principal operations of the algebra of incidence to compute the *intersection* and *union* of the k-planes. Suppose that the set of r points $a_1, a_2, \ldots, a_r \in \Pi^n$ and the set of s points $b_1, b_2, \ldots, b_s \in \Pi^n$ are both in general position (linearly independent vectors in $\mathbb{R}^{n+1}$), then the $(r-1)$-plane in Π^n is specified by the r-blade

$$A_r = a_1 \wedge a_2 \wedge \cdots \wedge a_r \neq 0, \tag{3.254}$$

and the $(s-1)$-plane by the s-blade

$$B_s = b_1 \wedge b_2 \wedge \cdots \wedge b_s \neq 0. \tag{3.255}$$

Considering the a's and b's to be the basis elements of respective subspaces $\mathcal{A}_r$ and $\mathcal{B}_s$, they can be sorted in such a way that

$$\mathcal{A}^r \cup \mathcal{B}^s = span\{a_1, a_2, \ldots, a_s, b_{\lambda_1}, \ldots, b_{\lambda_k}\}. \tag{3.256}$$

Supposing that

$$B_s = b_{\lambda_1} \wedge \cdots \wedge b_{\lambda_k} \wedge b_{\alpha_1} \wedge \cdots \wedge b_{\alpha_{s-k}}, \tag{3.257}$$

it follows that the "meet" and "join" of the r-blade A_r and s-blade B_s are, respectively, given by

$$A_r \cup B_s = A_r \wedge b_{\lambda_1} \wedge \cdots \wedge b_{\lambda_k}, \tag{3.258}$$

$$\mathcal{A}^r \cap \mathcal{B}^s = span\{b_{\alpha_1}, \ldots, b_{\alpha_{s-k}}\}. \tag{3.259}$$

Note that if the meet of A_r and $B_s = 0$, their join equals the wedge of the blades $A_r \cup B_s = A_r \wedge B_s$.

After the join of $\mathcal{A}^r$ and $\mathcal{B}^s$ has been computed, the $r+k$-blade

$$\overline{I}_{A_r \cup B_s} = \mathcal{A}^r \cup \mathcal{B}^s \tag{3.260}$$

can be used for computing the meet of the r- and s-blades A_r and B_s:

$$A_r \cap B_s = A_r \cdot (B_s \cdot I_{A_r \cup B_s}) = (I_{A_r \cup B_s} \cdot A_r) \cdot B_s. \tag{3.261}$$

This expression holds for the positive definite metric of $\mathbb{R}^{n+1}$. If we use any non-degenerated pseudo-Euclidean space $\mathbb{R}^{p,q}$, where $p+q = n+1$, we must use

instead the reciprocal $r+k$-blade $\overline{I}_{A_r \cup B_s}$, for which the property $I_{A_r \cup B_s} \cdot \overline{I}_{A_r \cup B_s} \neq 0$ is satisfied. For this case, the meet equation reads

$$A_r \cap B_s = A_r \cdot (B_s \cdot \overline{I}_{A_r \cup B_s}) = (\overline{I}_{A_r \cup B_s} \cdot A_r) \cdot B_s. \tag{3.262}$$

Note that if the grade of the blade $A_r \cup B_s$ equals $n = p + q$, we can simply use the inverse of the pseudoscalar, so that $I \cdot \overline{I} = 1$.

In the case of the geometric algebra of the null cone $G(N^{n+1})$, we define the following reciprocal $r+k$-blade for meet Eq. (3.262):

$$\overline{I}_{A_r \cup B_s} = \overline{a}_1 \wedge \overline{a}_2 \wedge \cdots \wedge \overline{a}_s \wedge \overline{b}_{\lambda_1} \wedge \cdots \wedge \overline{b}_{\lambda_k}. \tag{3.263}$$

A more complete discussion of these ideas can be found in [4, 15].

3.11.1 Incidence Relations in the Affine n-Plane

This subsection presents incidence relations between points, lines, planes, and higher dimensional k-planes using the useful computational framework of the affine n-plane. Let us rewrite Eq. (3.203) in the larger pseudo-Euclidean space $\mathbb{R}^{n+1,1} = \mathbb{R}^n \oplus \mathbb{R}^{1,1}$, where $\mathbb{R}^{1,1} = span\{e_{n+1}, \bar{e}_1\}$:

$$\mathcal{A}_e(\mathbb{R}^n) = \{x_h = x + e |\ x \in \mathbb{R}^n\} \subset \mathbb{R}^{n+1,1}. \tag{3.264}$$

The null vector $e \in \mathbb{R}^{1,1}$ is given by $e = \frac{1}{2}(e_{n+1} + \bar{e}_1)$, and the reciprocal null vector $\overline{e} = (e_{n+1} - \bar{e}_1)$ fulfills the condition $e \cdot \overline{e} = 1$. Now, if we merge the n-affine plane $\mathcal{A}_e(\mathbb{R}^n)$ together with the plane at infinity, we obtain the projective plane Π^n. Each point $\boldsymbol{x} \in \mathcal{A}_e(\mathbb{R}^n)$ is called a *homogeneous representative* of the corresponding point in Π^n. Now points in the affine plane can be represented as rays in the projective space:

$$\mathcal{A}_e^{rays}(\mathbb{R}^n) = \{\boldsymbol{y} |\ \ \boldsymbol{y} \in \mathbb{R}^{n+1} \text{ and } \boldsymbol{y} \cdot \bar{e} \neq 0\} \subset \mathbb{R}^{n+1}. \tag{3.265}$$

Note that in this definition we consider $\boldsymbol{y} \cdot \bar{e} \neq 0$, because rays are directions and they remain the same if we multiply for a scalar. Accordingly, a homogeneous point of the n-affine plane can be uniquely computed from a ray as follows:

$$\frac{\boldsymbol{y}}{\boldsymbol{y} \cdot \bar{e}} \in \mathcal{A}_e(\mathbb{R}^n). \tag{3.266}$$

Now let us formulate useful incidence relations. If we consider k-points $a_1^h, a_2^h, \ldots, a_k^h \in \mathcal{A}_e^n$, where each $a_i^h = a_i + e$ for $a_i \in \mathbb{R}^n$, and then compute their the outer product, we get the $(k-1)$-plane A^h in Π^n:

$$
\begin{aligned}
A^h &= a_1^h \wedge a_2^h \wedge \cdots \wedge a_k^h = a_1^h \wedge (a_2^h - a_1^h) \wedge a_3^h \wedge \cdots \wedge a_k^h = \cdots, \\
&= a_1^h \wedge (a_2^h - a_1^h) \wedge (a_3^h - a_2^h) \wedge \cdots \wedge (a_k^h - a_{k-1}^h), \\
&= a_1^h \wedge (a_2 - a_1) \wedge (a_3 - a_2) \wedge \cdots \wedge (a_k - a_{k-1}), \\
&= (a_1 + e) \wedge (a_2 - a_1) \wedge (a_3 - a_2) \wedge \cdots \wedge (a_k - a_{k-1}), \\
&= a_1 \wedge a_2 \wedge \cdots \wedge a_k + \\
&\qquad + e \wedge (a_2 - a_1) \wedge (a_3 - a_2) \wedge \cdots \wedge (a_k - a_{k-1}).
\end{aligned} \tag{3.267}
$$

This equation represents a $(k-1)$-plane in Π^n, but it also belongs to the affine n-plane $\mathcal{A}_e^n$ and thus contains important metrical information which can be extracted by taking the dot product from the left with $\overline{e}$:

$$
\begin{aligned}
\overline{e} \cdot A^h &= \overline{e} \cdot (a_1^h \wedge a_2^h \wedge \cdots \wedge a_k^h), \\
&= (a_2 - a_1) \wedge (a_3 - a_2) \wedge \cdots \wedge (a_k - a_{k-1}).
\end{aligned} \tag{3.268}
$$

Interestingly enough, this result, with a little modification, turns out to be the directed content of the $(k-1)$-simplex $A^h = a_1^h \wedge a_2^h \wedge \cdots \wedge a_k^h$ in the affine n-plane:

$$
\begin{aligned}
\frac{\overline{e} \cdot A^h}{(k-1)!} &= \frac{\overline{e} \cdot (a_1^h \wedge a_2^h \wedge \cdots \wedge a_k^h)}{(k-1)!}, \\
&= \frac{(a_2 - a_1) \wedge (a_3 - a_2) \wedge \cdots \wedge (a_k - a_{k-1})}{(k-1)!}.
\end{aligned} \tag{3.269}
$$

3.11.2 Directed Distances

Using our previous results, we can propose useful equations in the affine plane to relate points, lines, and planes metrically. The *directed distance* or *foot* from the $(k-1)$-plane $a_1^h \wedge \cdots \wedge a_k^h$ to the point b^h is given by

$$
\begin{aligned}
d[a_1^h \wedge \cdots a_k^h, b^h] &\equiv [\{\overline{e} \cdot (a_1^h \wedge \cdots \wedge a_k^h)\}\ (\overline{e} \cdot b^h)]^{-1} [\overline{e} \cdot (a_1^h \wedge \cdots \wedge a_k^h \wedge b^h)], \\
&= [a_2 - a_1) \wedge \cdots \wedge (a_k - a_{k-1})]^{-1} [(a_2 - a_1) \wedge \cdots \wedge (a_k - a_{k-1}) \wedge (b - a_k)].
\end{aligned} \tag{3.270}
$$

In the same sense, the equation of the directed distance between the two lines $a_1^h \wedge a_2^h$ and $b_1^h \wedge b_2^h$ in the affine n-plane reads

$$
\begin{aligned}
d[\ a_1^h \wedge a_2^h, b_1^h \wedge b_2^h] &\equiv [\{\overline{e} \cdot (a_1^h \wedge a_2^h)\} \wedge \{\overline{e} \cdot (b_1^h \wedge b_2^h)\}]^{-1} [\overline{e} \cdot (a_1^h \wedge a_2^h \wedge b_1^h \wedge b_2^h)] \\
&= [(a_2 - a_1) \wedge (b_2 - b_1)]^{-1} [(a_2 - a_1) \wedge (b_1 - a_2) \wedge (b_2 - b_1)].
\end{aligned} \tag{3.271}
$$

A general equation of the directed distance between the $(r-1)$-plane $A^h = a_1^h \wedge \cdots \wedge a_r^h$ and the $(s-1)$-plane $B^h = b_1^h \wedge \cdots \wedge b_s^h$ in the affine n-plane is similarly given by

$$\begin{aligned} &d[a_1^h \wedge \cdots \wedge a_r^h, b_1^h \wedge \cdots \wedge b_s^h] \equiv \\ &\{\bar{e} \cdot (a_1^h \wedge \cdots \wedge a_r^h)\} \wedge \{\bar{e} \cdot (b_1^h \wedge \cdots \wedge b_s^h)\}]^{-1} [\bar{e} \cdot (a_1^h \wedge \cdots \wedge a_r^h \wedge b_1^h \wedge \cdots \wedge b_s^h)], \\ &= [(a_2 - a_1) \wedge \cdots \wedge (a_r - a_{r-1}) \wedge (b_2 - b_1) \wedge \cdots \wedge (b_s - b_{s-1})]^{-1} \\ &[(a_2 - a_1) \wedge \cdots \wedge (a_r - a_{r-1}) \wedge (b_1 - a_r) \wedge (b_2 - b_1) \wedge \cdots \wedge (b_s - b_{s-1})]. \end{aligned} \tag{3.272}$$

We have to be careful, because if $A^h \wedge B^h = 0$, the directed distance may or may not be equal to zero. If $(a_1^h \wedge \cdots \wedge a_r^h) \wedge (b_1^h \wedge \cdots \wedge b_{s-1}^h) \neq 0$, we can calculate the meet between the $(r-1)$-plane A^h and $(s-1)$-plane B^h,

$$\begin{aligned} p &= (a_1^h \wedge \cdots \wedge a_r^h) \cap (b_1^h \wedge \cdots \wedge b_s^h), \\ &= (a_1^h \wedge \cdots \wedge a_r^h) \cdot [(b_1^h \wedge \cdots \wedge b_s^h) \cdot \overline{I}_{A \cup B}], \end{aligned} \tag{3.273}$$

where

$$\overline{I}_{A \cup B} = \{\bar{e} \cdot [(a_1^h \wedge \cdots \wedge a_r^h) \wedge (b_1^h \wedge \cdots \wedge b_{s-1}^h)]\} \wedge \bar{e}.$$

It can happen that the point $p = A^h \cap B^h$ may not be in the affine n-plane, but the *normalized* point $p^h = \frac{p}{\bar{e} \cdot p}$ will either be in the affine plane or will be undefined. Finding the "normalized point" is not necessary in many calculations but is required when the metric plays an important role or in the case of parallel hyperplanes, when it is used as an indicator.

3.11.3 Incidence Relations in the Affine 3-Plane

This subsection presents some algebra of incidence relations for 3D Euclidean space represented in the affine 3-plane $\mathcal{A}_e^3$, with the pseudoscalar $I = e_{123}e$ and the reciprocal pseudoscalar $\overline{I} = \bar{e}e_{321}$ satisfying the condition $I \cdot \overline{I} = 1$. Similar incidence relations were given by Blaschke [25] using dual quaternions, and later by Selig using the 4D degenerate geometric algebra $G_{3,0,1}$ [26]. Unlike the formulas given by these authors, our formulas are generally valid in any dimension and are expressed completely in terms of the meet and join operations in the affine plane. Blaschke and Selig could not exploit the meet and join operations because they were using a geometric algebra with a degenerate metric.

The distance of a point b^h to the line $L^h = a_1^h \wedge a_2^h$ is the *magnitude* or *norm* of the directed distance,

$$|d| = \left| \left[\{\bar{e} \cdot (a_1^h \wedge a_2^h)\} \wedge \{(\bar{e} \cdot (b^h)\} \right]^{-1} \left[\bar{e} \cdot (a_1^h \wedge a_2^h \wedge b^h) \right] \right|. \tag{3.274}$$

The distance of a point b^h to the plane $A^h = a_1^h \wedge a_2^h \wedge a_3^h$ is

$$|d| = \left| \left[\{\bar{e} \cdot (a_1^h \wedge a_2^h \wedge a_3^h)\} \wedge \{(\bar{e} \cdot (b^h)\} \right]^{-1} \left[\bar{e} \cdot (a_1^h \wedge a_2^h \wedge a_3^h \wedge b^h) \right] \right|. \quad (3.275)$$

Let us analyze carefully the incidence relation between the lines $L_1^h = a_1^h \wedge a_2^h$ and $L_2^h = b_1^h \wedge b_2^h$, which are completely determined by their join $I_{L_1^h \cup L_2^h} = L_1^h \cup L_2^h$. The following formulas help to test the incidence relations of the lines.

- If $I_{L_1^h \cup L_2^h}$ is a bivector, the lines coincide and $L_1^h = t L_2^h$ for some $t \in \mathbb{R}$.
- If $I_{L_1^h \cup L_2^h}$ is a 3-vector, the lines are either parallel or intersect in a common point. In this case,

$$p = L_1^h \cap L_2^h = L_1^h \cdot \left(L_2^h \cdot \overline{I}_{L_1^h \cup L_2^h} \right), \quad (3.276)$$

  where p is the result of the meet. If $\bar{e} \cdot p = 0$, the lines are parallel; otherwise, they intersect at the point $p_h = \frac{p}{\bar{e} \cdot p}$ in the affine 3-space $\mathcal{A}_e^3$.
- If $I_{L_1^h \cup L_2^h}$ is a 4-vector, the lines are skew. In this case, the distance is given by Eq. (3.271).

The incidence relation between a line $L^h = a_1^h \wedge a_2^h$ and a plane $B^h = b_1^h \wedge b_2^h \wedge b_3^h$ is also determined by their join, $L^h \cup B^h$. Clearly, if the join is a trivector, the line L^h lies in the plane B^h. The only other possibility is that their join is the pseudoscalar $I = \sigma_{123} e$. In this case,

$$p = L^h \cap B^h = L^h \cdot (B^h \cdot \overline{I}). \quad (3.277)$$

If $\bar{e} \cdot p = 0$, the line is parallel to the plane, with the directed distance determined by Eq. (3.272). Otherwise, their point of intersection in the affine plane is $p_h = \frac{p}{\bar{e} \cdot p}$.

Two planes, $A^h = a_1^h \wedge a_2^h \wedge a_3^h$ and $B^h = b_1^h \wedge b_2^h \wedge b_3^h$, in the affine plane $\mathcal{A}_e^3$ are either parallel, intersect in a line, or coincide. If their join is a trivector, that is, if $A^h = t B^h$ for some $t \in \mathbb{R}^*$, they obviously coincide. If they do not coincide, then their join is the pseudoscalar $I = \sigma_{123} e$. In this case, we calculate the meet as

$$L = A^h \cap B^h = (\overline{I} \cdot A^h) \cdot B^h. \quad (3.278)$$

If $\bar{e} \cdot L = 0$, the planes are parallel, with the directed distance determined by Eq. (3.272). Otherwise, L represents the line of intersection in the affine plane having the direction $\bar{e} \cdot L$.

The equivalent of the above incidence relations was given by Blaschke [25] using dual quaternions, and by Selig [27] utilizing a special or degenerate four-dimensional Clifford algebra. Whereas Blaschke uses only pure quaternions (bivectors) for his representation, Selig uses trivectors for points and vectors for planes. In contrast, in the affine 3-plane, points are always represented by vectors, lines by bivectors, and

planes by trivectors. This offers a comprehensive and consistent interpretation which greatly simplifies the underlying conceptual framework. The following equations compare our equations (left side) with those of Blaschke and Selig (right side).

$$\text{Equation (3.274)} \equiv \frac{1}{2}(\tilde{p}l + \tilde{l}p), \tag{3.279}$$

$$\text{Equation (3.275)} \equiv \frac{1}{2}(\tilde{p}\pi + \tilde{\pi}p), \tag{3.280}$$

$$\text{Equation (21.11)} \equiv \frac{1}{2}(\tilde{l}\pi + \tilde{\pi}l). \tag{3.281}$$

3.11.4 Geometric Constraints as Flags

It is often necessary to check a geometric configuration during a rigid motion in Euclidean space, and simple geometric incidence relations can be used for this purpose. For example, a point p is on a line L if and only if

$$p \wedge L = 0. \tag{3.282}$$

Similarly, a point p is on a plane A if

$$p \wedge A = 0. \tag{3.283}$$

A line L will lie in plane A if

$$L \cap A = A. \tag{3.284}$$

Alternatively, the line L can meet the plane A in a single point p, in which case,

$$L \cap A = p,$$

or, if the line L is parallel to the plane A,

$$L \cap A = 0. \tag{3.285}$$

3.12 Conclusion

We have shown how geometric algebra can effectively be used to carry out analysis on a manifold, which is useful in robotics and image analysis. Geometric algebra offers a clear and concise geometric framework of multivectors in which calculations can be carried out. Since the elements and operations in geometric algebra are basis

robot free, computations are simpler and geometrically more transparent than in more traditional approaches.

Stereographic projection and its generalization to the conformal group and projective geometry have direct application to image analysis from one or more viewpoints. The key idea is that an image is first represented on the null cone, and then projected onto affine geometries or onto an n-dimensional affine plane, where the image analysis takes place. Since every Lie algebra can be represented by an appropriate bivector algebra in an affine geometry, it follows that a complete motion analysis should be possible using its bivector representation in geometric algebra.

3.13 Exercises

3.1 A complex structure is introduced through the doubling bivector,

$$\boldsymbol{J} = e_1 \wedge f^1 + e_2 \wedge f^2 + e_3 \wedge f^3 + \cdots + e_n \wedge f^n = e_i \wedge f^i,$$

if the $\{e_i\}$ frame is chosen to be orthonormal, we obtain

$$\boldsymbol{J} = e_1 f_1 + e_2 f_2 + e_3 f_3 + \cdots + e_n f_n = e_i f_i = \boldsymbol{J}_1 + \boldsymbol{J}_2 + \cdots + \boldsymbol{J}_n.$$

This sum consists of n commuting blades of grade two. Give the geometric interpretation of each bivector $\boldsymbol{J}_i$. What represents each $\boldsymbol{J}_i$ w.r.t. to the rotation plane expressed by $\boldsymbol{J}$.

3.2 The vector $\{e_i\}, i = 1, \ldots, 3$, is the basis for G_3. The bivector algebra of the rotor algebra G_3^+ is spanned by the bivectors e_{23}, e_{31}, e_{12}. Prove that this algebra fulfills the axiom of closure under the commutator product. Find the structure constants of the unitary group as well.

3.3 The bivector algebra of the rotor algebra G_3^+ is spanned by the bivectors e_{23}, e_{31}, e_{12}. Compute the Killing of this bivector algebra.

3.4 The vector $\{e_i, f_i\}, i = 1, \ldots, n$, is the basis for $G_{2n,0}$. The Lie algebra $u(n)$ is defined by the following bivectors:

$$\begin{aligned} J_i &= e_i f_i && (i = 1, \ldots, n), \\ E_{ij} &= e_i e_j + f_i f_j && (i < j = 1, \ldots, n), \\ F_{ij} &= e_i f_j - f_i e_j && (i < j = 1, \ldots, n), \end{aligned}$$

for $i < j = 1, \ldots, n$. Prove that this algebra fulfills the axiom of closure under the commutator product. Find the structure constants of the unitary group as well.

3.5 Compute the Killing of the Lie algebra $u(n)$ defined by the bivectors given in Exercise 3.3.

3.6 Prove that the quaternion algebra $\mathbb{H}$ isomorphic to the rotor algebra G_3^+ is a ring.

3.7 The $\{e_i\}$ basis vectors are orthonormal with positive signature, and the $\{\bar{e}_i\}$ are orthonormal with negative signature and belong to the geometric algebra $G_{n,n,}$. By using the constraint of Eq. (3.114), given by

$$[\boldsymbol{x} \wedge \boldsymbol{y} - (\boldsymbol{x} \cdot \boldsymbol{K})(\boldsymbol{y} \cdot \boldsymbol{K})] \times \boldsymbol{K} = 0, \tag{3.286}$$

one can again try all combinations of $\{e_i, \bar{e}_i\}$ to produce the bivector basis for the Lie algebra $gl(n)$ of the general linear group $GL(n)$,

$$\begin{aligned} \boldsymbol{K}_i &= \frac{1}{2}\boldsymbol{F}_{ij} = e_i\bar{e}_i, \\ \boldsymbol{E}_{ij} &= e_ie_j - \bar{e}_i\bar{e}_j \qquad (i < j = 1, \ldots, n), \\ \boldsymbol{F}_{ij} &= e_i\bar{e}_j - \bar{e}_ie_j \qquad (i < j = 1, \ldots, n), \end{aligned}$$

for $i < j = 1, \ldots, n$. Prove that this algebra fulfills the axiom of closure under the commutator product. Find the structure constants of the unitary group as well.

3.8 The $\{e_i\}$ basis vectors are orthonormal with positive signature, and the $\{\bar{e}_i\}$ are orthonormal with negative signature and belong to the geometric algebra $G_{n,n,}$. Since $\underline{K}_*$ defines a new Lie algebra with generators determined by the outermorphism condition

$$\underline{K}_*(\boldsymbol{B}) = -\boldsymbol{B},$$

using this result, one can construct a generator basis for an invariance group of $\underline{K}_*$ as follows:

$$\begin{aligned} \boldsymbol{E}_{ij} &= e_ie_j - \bar{e}_i\bar{e}_j, \\ \boldsymbol{F}_{ij} &= e_i\bar{e}_j + \bar{e}_ie_j, \end{aligned}$$

for $i, j = 1, 2, \ldots, n, i < j$. These are the generators of the *complex orthogonal group* $\mathrm{SO}(n, \mathbb{C})$. Prove that this Lie algebra fulfill the axiom of closure under the commutator product. Find the structure constants of the unitary group as well. Explain also why for odd n, $\boldsymbol{K}$ is the only kind of involutory bivector. When the case n is even, show which other groups are determined by their invariants.

3.9 Prove how the versor for translation called translator, $\boldsymbol{T} \in G_{3,3}$, is derived in terms of $e_i\bar{e}_j$ and $w_i \wedge \bar{w}_j$ as follows:

$$\begin{aligned}\boldsymbol{T} &= e^{\boldsymbol{B}_t} = e^{\frac{t_3\bar{e}_1e_2 - t_2\bar{e}_1e_3 + t_1\bar{e}_2e_3}{2}},\\ &= e^{\frac{t_3\bar{w}_1\wedge\bar{w}_2 - t_2\bar{w}_1\wedge\bar{w}_3 + t_1\bar{w}_2\wedge\bar{w}_3}{2}},\\ &= 1 + \frac{1}{2}(t_3\bar{e}_1e_2 - t_2\bar{e}_1e_3 + t_1\bar{e}_2e_3),\\ &= 1 + \frac{1}{2}(t_3\bar{w}_1\wedge w_2 - t_2\bar{w}_1\wedge\bar{w}_3 + t_1\bar{w}_2\wedge\bar{w}_3).\end{aligned}$$

3.10 Prove how the versor for perspectivity called perspector, $\boldsymbol{P} \in G_{3,3}$, is derived in terms of $e_i\bar{e}_j$ and $w_i \wedge \bar{w}_j$ as follows:

$$\begin{aligned}\boldsymbol{P} &= e^{\boldsymbol{B}_f} = e^{\frac{-f_3e_1\bar{e}_2 + f_2e_1\bar{e}_3 - f_1e_2\bar{e}_3}{2}},\\ &= e^{\frac{-f_3w_1\wedge w_2 + f_2w_1\wedge w_3 - f_1w_2\wedge w_3}{2}},\\ &= 1 + \frac{1}{2}(-f_3e_1\bar{e}_2 + f_2e_1\bar{e}_3 - f_1e_2\bar{e}_3),\\ &= 1 + \frac{1}{2}(-f_3w_1\wedge w_2 + f_2w_1\wedge w_3 - f_1w_2\wedge w_3).\end{aligned}$$

3.11 Prove how the versor for rotation called rotor, $\boldsymbol{R} \in G_{3,3}$, is derived in terms of $e_i\bar{e}_j$ and $w_i \wedge \bar{w}_j$ as follows:

$$\begin{aligned}\boldsymbol{R} &= e^{\boldsymbol{B}_R},\\ &= e^{\frac{(-e_2e_3 + \bar{e}_2\bar{e}_3) + (-e_1e_3 + \bar{e}_1\bar{e}_3) + (-e_1e_2 + \bar{e}_1\bar{e}_2)}{2}},\\ &= e^{\frac{(w_2\wedge\bar{w}_3 - \bar{w}_2\wedge w_3) + (w_1\wedge\bar{w}_3 - \bar{w}_1\wedge w_3) + (w_1\wedge\bar{w}_2 - \bar{w}_1\wedge w_2)}{2}}.\end{aligned}$$

3.12 The bivector $\boldsymbol{B}_{\theta_{23}} = \frac{1}{2}(w_2 \wedge \bar{w}_3 - \bar{w}_2 \wedge w_3) = \frac{1}{2}(-e_2e_3 + \bar{e}_2\bar{e}_3)$ squares to $\boldsymbol{B}^2_{\theta_{23}} = -\frac{1}{2}(1 + w_2 \wedge \bar{w}_3 \wedge \bar{w}_2 \wedge w_3)$. Since $\boldsymbol{B}^3_{\theta_{23}} = -\boldsymbol{B}_{\theta_{23}}$ proof that

$$e^{\theta\boldsymbol{B}_{\theta_{23}}} = 1 + \sin(\theta)\boldsymbol{B}_{\theta_{23}} + (1 - \cos(\theta))\boldsymbol{B}^2_{\theta_{23}}. \tag{3.287}$$

See in Appendix B the equations of the exponentials of bivectors.

3.13 In Sect. 3.10.3, we derived for $Spin(2)$ the equation

$$\begin{aligned}\boldsymbol{R}_\theta = \exp\left(\frac{1}{2}\theta\boldsymbol{B}\right) = \exp\left(\frac{0\cdot\theta}{2}\right)p_1 + \exp(i\theta)p_2 + \exp(-i\theta)p_3,\\ = p_1 + \cos(\theta)(p_2 + p_3) + \sin(\theta)i(p_2 - p_3),\\ = p_1 + \cos(\theta)(p_2 + p_3) + \frac{\boldsymbol{B}}{2}\sin(\theta).\end{aligned}$$

Now, similar as in Exercise 3.12, for the rotor

$$\begin{aligned}\boldsymbol{R} &= e^{\boldsymbol{B}_R},\\ &= e^{\frac{(-e_2e_3 + \bar{e}_2\bar{e}_3) + (-e_1e_3 + \bar{e}_1\bar{e}_3) + (-e_1e_2 + \bar{e}_1\bar{e}_2)}{2}},\\ &= e^{\frac{(w_2\wedge\bar{w}_3 - \bar{w}_2\wedge w_3) + (w_1\wedge\bar{w}_3 - \bar{w}_1\wedge w_3) + (w_1\wedge\bar{w}_2 - \bar{w}_1\wedge w_2)}{2}},\end{aligned}$$

write $\boldsymbol{R}$ in terms of trigonometric functions. See in Appendix B the equations of the exponentials of bivectors.

3.14 Prove how the versor for Lorentz transformation called Lorentor, $\boldsymbol{R}_L \in G_{3,3}$, is derived in terms of $e_i\bar{e}_j$ and $w_i \wedge \bar{w}_j$ as follows:

$$\begin{aligned}\boldsymbol{R}_L &= e^{\boldsymbol{B}_L},\\ &= e^{\frac{(e_2\wedge e_3-\bar{e}_2\wedge\bar{e}_3)+(e_1\wedge e_3-\bar{e}_1\wedge\bar{e}_3)+(e_1\wedge e_2-\bar{e}_1\wedge\bar{e}_2)}{2}},\\ &= e^{\frac{(\bar{w}_2\wedge w_3-w_2\wedge\bar{w}_3)+(\bar{w}_1\wedge w_3-w_1\wedge\bar{w}_3)+(\bar{w}_1\wedge w_2-w_1\wedge\bar{w}_2)}{2}}.\end{aligned}$$

3.15 Prove how the versor for shear affine transformation called Sheartor, $\boldsymbol{S} \in G_{3,3}$, is derived in terms of $e_i\bar{e}_j$ and $w_i \wedge \bar{w}_j$ as follows:

$$\begin{aligned}\boldsymbol{S} &= e^{\boldsymbol{B}_S},\\ &= e^{\frac{a_{23}(e_2\wedge e_3-\bar{e}_3\wedge\bar{e}_2)+a_{13}(e_1\wedge e_3-\bar{e}_3\wedge\bar{e}_1)+a_{12}(e_1\wedge e_2-\bar{e}_2\wedge\bar{e}_1)}{2}}\\ &\qquad e^{\frac{a_{32}(e_3\wedge e_2-\bar{e}_2\wedge\bar{e}_3)+a_{31}(e_3\wedge e_1-\bar{e}_1\wedge\bar{e}_3)+a_{21}(e_2\wedge e_1-\bar{e}_1\wedge\bar{e}_2)}{2}},\\ &= e^{\frac{a_{23}(\bar{w}_2\wedge w_3-w_3\wedge\bar{w}_2)+a_{13}(\bar{w}_1\wedge w_3-w_3\bar{w}_1)+a_{12}(\bar{w}_1\wedge w_2-w_2\wedge\bar{w}_1)}{2}}\\ &\qquad e^{\frac{a_{32}(\bar{w}_3\wedge w_2-w_2\wedge\bar{w}_3)+a_{31}(\bar{w}_3\wedge w_1-w_1\bar{w}_3)+a_{21}(\bar{w}_2\wedge w_1-w_1\bar{w}_2)}{2}}.\end{aligned}$$

3.16 Prove how the versor for dilation called dilator, $\boldsymbol{D} \in G_{3,3}$, is derived in terms of $e_i\bar{e}_j$ and $w_i \wedge \bar{w}_j$ as follows:

$$\begin{aligned}\boldsymbol{D} &= e^{\boldsymbol{B}_D},\\ &= e^{\frac{1}{2}\{(1/0,\pm)\bar{e}_1\wedge e_1+(0/1,\mp)\bar{e}_2\wedge e_2+(0/1,\mp)\bar{e}_3\wedge e_3\}}\\ &\qquad e^{\frac{1}{2}\{(-1/0,\pm)e_1\wedge\bar{e}_1+(0/-1,\mp)e_2\wedge\bar{e}_2+(0/-1,\mp)e_3\wedge\bar{e}_3\}},\\ &= e^{\frac{1}{2}\{(1/0,\pm)\bar{w}_1\wedge w_1+(0/1,\mp)\bar{w}_2\wedge w_2+(0/1,\mp)\bar{w}_3\wedge w_3\}}\\ &\qquad e^{\frac{1}{2}\{(-1/0,\pm)w_1\wedge\bar{w}_1+(0/-1,\mp)w_2\wedge\bar{w}_2+(0/-1,\mp)w_3\wedge\bar{w}_3\}}.\end{aligned} \tag{3.288}$$

3.17 Following a similar methodology for the versors obtained for the 3D projective geometry in $\mathbb{P}^3$ and using Eqs. (3.201)–(3.202), derive the versors for the Lie groups of $\mathbb{P}^2$, namely, for the translation (2-DOF), rotation (1-DOF) about the axis of the plane $e_1 \wedge e_2$, scaling (2-DOF), perspectivity (2-DOF), and shear (1-DOF).

3.18 Considering the camera pinhole model [14], the projective transformation from the projective space $\mathbb{P}^3$ to the projective plane $\mathbb{P}^2$ is decomposed in a multiplication of an affine transformation K a 3×3 upper triangular matrix representing the intrinsic parameters, and a 3×4 matrix which represents extrinsic parameters of the rigid motion $SE(3)$

$$P_{3\times 4} = K[R|t]. \tag{3.289}$$

Using Eq. (3.66) express the entries of matrix in terms of bivectors and then formulate the versor corresponding to the affine transformation of matrix K in the projective plane $\mathbb{P}^2$. Hint: Use the results of Exercise 3.15.

3.19 In Exercise 3.16, you have computed the versor for the matrix K; the relation of pairs of points in two views is established by the homography H as follows:

$$x_i' = Hx_i, \tag{3.290}$$

for any pair of points $x_i \in \mathbb{P}^2$. Now H can be decomposed as follows:

$$H = K[r_1, r_2, t], \tag{3.291}$$

where r_1, r_2 are the column vectors of the rotation matrix R, i.e., $r_3 = r_1 \times r_2$ and t is the translation vector in $P_{3\times 4} = K[R|t]$, where $t \cdot r_i = 0$, for $i = 1, 2, 3$. Using Eq. (3.66), express the entries of the matrix H in terms of bivectors and then formulate the versor for the homography H in the projective plane $\mathbb{P}^2$. Hint: Use the results of Exercise 3.15.

3.20 Prove that $\boldsymbol{x}_c = \frac{1}{2}\boldsymbol{x}_h\overline{w}\boldsymbol{x}_h = exp(\frac{1}{2}\boldsymbol{x}\overline{w})\ e\ exp(-\frac{1}{2}\boldsymbol{x}\overline{w})$, where $\boldsymbol{x}_c \in \mathcal{H}_w^{p,q}, x_h \in \mathcal{A}_w^{p,q}$, and $\boldsymbol{x} \in \mathbb{R}^n$.

3.21 The bases of the reciprocal null cones $\{w\} \in N$ and $\{\bar{w}\} \in N$ are called reciprocal or dual bases because they fulfill the relationship $\{w\} \cdot \{\bar{w}\} = id$, where id is an $n \times n$ identity matrix. The pseudoscalar of $G(N)$ is $I = w_1 \wedge w_2 \wedge w_3 \ldots w_n$, and of $G(\bar{N})$ is $\bar{I} = \bar{w}_1 \wedge \bar{w}_2 \wedge \bar{w}_3 \ldots \bar{w}_n$, both of which satisfy the condition $I \cdot \bar{I} = 1$. According to Eq. (3.57), we can express a second basis $\{a\} = \{w\}\mathcal{A} = \{w_1, w_2, w_3, \ldots, w_n\}A \in N$, where the matrix A is responsible for this change of basis. The hypervolume spanned by the basis $\{a\}$ is $\bigwedge_{i=1}^{n}\{a\} = a_1 \wedge a_2 \wedge \ldots \wedge a_n = \det(\mathcal{A})w_1 \wedge w_2 \wedge \ldots w_n = \det(\mathcal{A})\mathrm{I}$. The *bracket* of $\mathcal{A}$ is simply computed by taking the dot product of this hypervolume and the reciprocal pseudoscalar $\det(\mathcal{A}) = \bigwedge_{i=1}^{n}\{a\} \cdot \bar{I}$. Similar to the standard approach for obtaining a reciprocal basis, it is easy to see that the new reciprocal basis $\{\bar{a}\}$ can be computed by means of the equation

$$\overline{a}_i = (-1)^{i+1}\frac{(a_1 \wedge \cdots \wedge (1)_i \wedge \cdots \wedge a_n) \cdot \overline{I}}{[a_1\ a_2\ \cdots\ a_n]},$$

where a_i is left out of the wedge operation in position $(1)_i$. This expression guarantees that $\{\bar{a}\} \cdot \{a\} = id$. Find an expression for computing the inverse of the matrix $(\mathcal{A})$. (*Hint*: Use $\{\bar{a}\} = \mathcal{B}\{\bar{a}\}$.)

3.22 Projections from $G_{3,3}$ to $G_{3,0}$: Consider the Lie algebra $so(3)$ in $G_{3,3}$:

$$\mathcal{L}_x = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \mathcal{L}_y = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{bmatrix}, \mathcal{L}_z = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}.$$

Using CLICAL or eClifford, represent this Lie algebra in $G_{3,3}$ using the bivector matrices $\boldsymbol{L}_x$, $\boldsymbol{L}_y$, and $\boldsymbol{L}_z$. Take their projections $P(\boldsymbol{L}_x) = I^{-1}(I \cdot \boldsymbol{L}_x)$, $P(\boldsymbol{L}_y) = I^{-1}(I \cdot \boldsymbol{L}_y)$, and $P(\boldsymbol{L}_z) = I^{-1}(I \cdot \boldsymbol{L}_z)$ using the Euclidean pseudoscalar $I = e_1e_2e_3$ and also using the reciprocal pseudoscalar $I = \bar{e}_1\bar{e}_2\bar{e}_3$. Explain the dual relation of the results.

3.23 Using CLICAL or eClifford, compute in the 2D affine plane $\mathcal{A}_e^2$ the new position of the point $\boldsymbol{x} = 4e_1 + 2e_2 \in \mathcal{A}_w(\mathbb{R}^2)$ after the translation $\boldsymbol{t} = 6e_1 + 5e_2 \in \mathcal{A}_e(\mathbb{R}^2)$.

3.24 Using CLICAL or eClifford, compute in the 2D affine plane $\mathcal{A}_e^2$ the dilation of the point $\boldsymbol{x} = 3e_1 + 5e_2\mathbb{R}^2$ for

$$\mathcal{D}_u = \begin{bmatrix} e^u & 0 & 0 \\ 0 & e^u & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1.75 & 0 & 0 \\ 0 & 1.75 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

3.25 Using CLICAL or eClifford, compute in the geometric algebra of the null cone $G(N^2)$ the new position of the point $x_0 = 2e_1 + 3e_2 \in \mathcal{A}(\mathbb{R}^2)$ after a rotation of $\theta = \frac{\pi}{6}$. Use Eq. (3.227) with the bivector of the spinor group $Spin(2)$ $\boldsymbol{B} = -w_1 \wedge \bar{w}_2 + -w_2 \wedge \bar{w}_1$. Note that the rotation is not computed with the exponential function but rather with a function depending on the mutually annihilating idempotents.

3.26 Compute in the affine 2-plane $\mathcal{A}_e^2$, with the pseudoscalar $I = e_1 \wedge e_2 \wedge e$ and the reciprocal pseudoscalar $\overline{I} = e_1 \wedge e_2 \wedge \bar{e}$, the meet of the lines $L_1^h = a_1^h \wedge a_2^h$, and $L_2^h = b_1^h \wedge b_2^h$, where $a_1^h = 4e_1 + e$, $a_2^h = 2e_2 + e$, $b_1^h = e$, and $b_2^h = 2e_1 + 3e_2 + e$.

3.27 In the affine 2-plane $\mathcal{A}_e^2$, compute the intersecting point p^h of the lines L_1^h and L_3^h, where L_1^h is the line determined in problem 3.7 and L_3^h passes through the point $c_2^h = 4e_1 + 3e_2 + e$ and is orthogonal to the line L_1^h. (*Hint*: Consider the line $L_3^h = p^h \wedge c_2^h$ with the point $p^h = c_2^h + s\, i\,(a_1 - a_2)$ for $s \in \mathbb{R}$, where $i = e_1e_2$.) Note that $s \neq 0$ can be overlooked, because the line is uniquely defined by the 2-direction of the bivector $p^h \wedge c_2^h$ and not by its magnitude.

3.28 Theorem proving: Let a circle entered at the origin and $\boldsymbol{a}$ and $\boldsymbol{b}$ be the endpoints of the diameter. Take any point $\boldsymbol{c}$ on the circle and show in the 2-plane $\mathcal{A}_e^2$ that the lines $\boldsymbol{l}_{ac}$ and $\boldsymbol{l}_{cb}$ are perpendicular.

3.29 Theorem proving: Prove the theorem of Desargues's configuration in the 3D-projective plane Π^3. Consider that x_1, x_2, x_3 and y_1, y_2, y_3 are the vertices of two triangles in Π^3 and suppose that $(x_1 \wedge x_2) \cap (y_1 \wedge y_2) = z_3$, $(x_2 \wedge x_3) \cap (y_2 \wedge y_3) = z_1$, and $(x_3 \wedge x_1) \cap (y_3 \wedge y_1) = z_2$. You can claim that $c_1 \wedge c_2 \wedge c_3 = 0$ if and only if there is a point p such that $x_1 \wedge y_1 \wedge p = 0 = x_2 \wedge y_2 \wedge p = x_3 \wedge y_3 \wedge p$. (*Hint*: Express the point as linear combinations of a_1, b_1, a_2, b_2, and a_3, b_3. The other half of the proof follows by duality of the classical projective geometry.)

3.30 Theorem proving: Consider an arbitrary circumcircled triangle, see Fig. 3.3. From a point $\boldsymbol{d}$ on the circumcircle, draw three perpendiculars to the triangle sides $\boldsymbol{bc}$, $\boldsymbol{ca}$, and $\boldsymbol{ab}$ to meet the circle at points $\boldsymbol{a}_1$, $\boldsymbol{b}_1$, and $\boldsymbol{c}_1$, respectively.

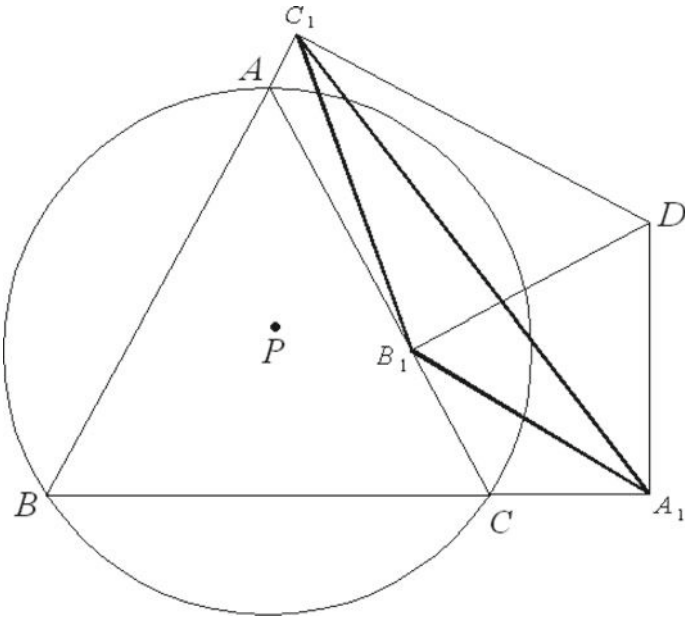


Fig. 3.3 Simson's theorem

Prove that the lines $\boldsymbol{l}_{aa_1}, \boldsymbol{l}_{bb_1}$, and $\boldsymbol{l}_{cc_1}$ are parallel. (*Hint*: In the affine 2-plane $\mathcal{A}_e^2$, interpret the geometry of your results according to the grade and the absolute value of the directed distances between the lines.)

3.31 Consider in the affine 3-plane $\mathcal{A}_e^3$ the points $a_1^h = 3e_1 + 4e_2 + 5e_3 + e$, $a_2^h = 2e_1 - 5e_2 + 2e_3 + e$, and $a_3^h = 1e_1 + 6e_2 + 4e_3 + e$; the line $L_1^h = a_1^h \wedge a_2^h$; and the plane $\phi_1^h = a_1^h \wedge a_2^h \wedge a_3$. Compute, using the Maple packet CLIFFORD 4.0, for (a) one of the points, (b) the line, and (c) the plane, their new positions after the rigid motion. This is defined by a translation of $t_1^h = e_1 + 2e_2 + e_3 + e$ and rotations about the three axes of $\theta_x = \frac{\pi}{5}$, $\theta_y = \frac{\pi}{3}$ and $\theta_z = \frac{\pi}{6}$. Recall that you compute the translation using the horosphere as an intermediate framework.

3.32 Consider in the affine 3-plane $\mathcal{A}_e^3$ the points $p_0^h = e$, $p_1^h = e_2 + e$, $p_2^h = e_1 + e_2 + e$, $p_3^h = e_1 + e$, $p_4^h = e_1 + e_3 + e$, $p_5^h = e_1 + e_2 + e_3 + e$, $p_6^h = e_2 + e_3 + e$, and $p_7^h = e_3 + e$; the lines $L_{01} = p_0^h \wedge p_1^h$, $L_{36} = p_3^h \wedge p_6^h$, and $L_{76} = p_7^h \wedge p_6^h$; and the planes $\phi_f = p_0^h \wedge p_1^h \wedge p_3^h$, $\phi_t = p_5^h \wedge p_6^h \wedge p_7^h$, and $\phi_r = p_2^h \wedge p_5^h \wedge p_6^h$. Compute, using CLIFFORD 4.0, the directed distances between p_7^h and ϕ_f, p_5^h and L_{36}, L_{01} and L_{36}, L_{36} and L_{76}, L_{01} and ϕ_t, L_{36} and ϕ_t, ϕ_f and ϕ_t, and ϕ_f and ϕ_r. Interpret the geometry of your results according to the grade and the absolute value of the directed distances.

References

1. Hestenes, D., & Sobczyk, G. (1984). *Clifford algebra to geometric calculus: A unified language for mathematics and physics*. Dordrecht: D. Reidel.
2. Doran, C., Hestenes, D., Sommen, F., & Van Acker, N. (1993). Lie groups and spin groups. *Journal of Mathematical Physics*, *34*(8), 3642–3669.
3. Doran, C., & Lasenby, A. (2005). *Geometric algebra for physicists*. Cambridge University Press.
4. Sobczyk, G. (2001). Universal geometric algebra. In E. Bayro-Corrochano & G. Sobczyk (Eds.), *Advances in geometric algebra with applications in science and engineering* (pp. 18–41). Boston: Birkhäuser.
5. Sommen, F., & Van Acker, N. (1993). SO(m)-invariant differential operators on Clifford algebra-valued functions. *Foundations of Physics*, *23*(11), 1491–1519.
6. Doran, C. (1994). *Geometric algebra and its applications to mathematical physics*. Ph.D. Thesis, University of Cambridge.
7. Li, H., Huang, L., Shao, Ch., & Dong, L. (2015). Three-dimensional projective geometry with geometric algebra. arXiv:1507.06634v1.
8. Bayro-Corrochano, E. (2001). *Geometric computing for perception action systems*. New York: Springer-Verlag.
9. Bayro-Corrochano, E., Daniilidis, K., & Sommer, G. (2000). Motor algebra for 3D kinematics. The case of the hand–eye calibration. In *International Journal of Mathematical Imaging and Vision*, *13*(2), 79–99.
10. Bayro-Corrochano, E. (2010). *Geometric computing for wavelet transforms, robot vision, learning, control and action*. London: Springer-Verlag.
11. Li, H., & Zhang, L. (2011). Line geometry in terms of the null geometric algebra over $\mathbb{R}^{3,3}$, and applications to the inverse singularity of generalized Stewart plataforms. In L. Dorst & J. Lasenby (Eds.), *Guide to geometric algebra in practice* (pp. 253–272). New York: Springer-Verlag.
12. Klawitter, D. (2014). A Clifford algebraic approach to line geometry. *Advances in Applied Clifford Algebra*, *24*, 713–736.
13. Dorst, L. (2015). 3D Projective geoemtry through versors of $\mathbb{R}^{3,3}$. *Advances in Applied Clifford Algebras*, *26*, 1137–1172.
14. Hartley, R. I., & Zissermann, A. (2003). *Multiple view geometry in computer vision* (2nd ed.). Cambridge: Cambridge University Press.
15. Pozo, J. M., & Sobczyk, G. (2001). Realizations of the conformal group. In E. Bayro-Corrochano & G. Sobczyk, (Eds.), *Geometric algebra with applications in science and engineerings* (pp. 42–60). Boston: Birkhäuser.
16. Belinfante, J., & Kolman, B. (1972). *Lie groups and lie algebras: With applications and computational methods*. Philadelphia: SIAM.
17. Fulton, W., & Harris, J. (1991). *Representation theory: A first course*. New York: Springer-Verlag.
18. Sobczyk, G. (1997). The generalized spectral decomposition of a linear operator. *The College Mathematics Journal*, *28*(1), 27–38.
19. Riesz, M. (1958). *Clifford numbers and spinors*. Lecture Series no. 38, University of Maryland. Reprinted as facsimile, E.F. Bolinder & P. Lounesto, (Eds.) (1993). Dordrecht: Kluwer.
20. Miller, W. (1968). *Lie theory and special functions*. New York: Academic Press.
21. Sobczyk, G. (1997). Spectral integral domains in the classroom. *Aportaciones Matemáticas, Serie Comunicaciones*, *20*, 169–188.
22. Lounesto, P. (1987). *CLICAL software packet and user manual*. Helsinki University of Technology of Mathematics, Research report A248.
23. Li, H., Hestenes, D., & Rockwood, A. (2001). Generalized homogeneous coordinates for computational geometry. In G. Sommer (Ed.), *Geometric computing with Clifford algebra* (pp. 27–52). New York: Springer-Verlag.

24. Hoffman, W. C. (1966). The Lie algebra of visual perception. *Journal of Mathematical Psychology*, *3*, 65–98.
25. Blaschke, W. (1960). *Kinematik und Quaternionen*. Berlin: VEB Deutscher Verlag der Wissenschaften.
26. Selig, J. (2000). Robotics kinematics and flags. In E. Bayro-Corrochano & G. Sobczyk (Eds.), *Advances in geometric algebra with applications in science and engineering* (pp. 211–234). Boston: Birkhäuser.
27. Selig, J. M. (1999). Clifford algebra of points, lines and planes. Technical report SBU-CISM-99-06, South Bank University, School of Computing, Information Technology and Maths.

Chapter 4
2D, 3D, and 4D Geometric Algebras

It is the belief that imaginary numbers appeared for the first time around 1540 when the mathematicians Tartaglia and Cardano represented real roots of a cubic equation in terms of conjugated complex numbers. A Norwegian surveyor, Caspar Wessel, was in 1798 the first one to represent complex numbers by points on a plane with its vertical axis imaginary and horizontal axis real. This diagram was later known as the Argand diagram, although the true Aragand's achievement was an interpretation of $i = \sqrt{(-1)}$ as a rotation by a right angle in the plane. Complex numbers received their name by Gauss, and their formal definition as pair of real numbers was introduced by Hamilton in 1835.

4.1 Complex, Double, and Dual Numbers

In a broad sense, the most general complex numbers [1, 2] on the plane can be categorized into three different systems: ordinary complex numbers, double numbers, and dual numbers. In general, a complex number can be represented as a *composed number* $\mathbf{a} = b + \omega c$ using the algebraic operator ω, in which $\omega^2 = -1$ in the case of complex numbers, $\omega^2 = 1$ in the case of double numbers, and $\omega^2 = 0$ in the case of dual numbers. For dual numbers, b represents the real term and c the dual term.

For this book, it is useful to recall the notion of a function of a dual variable, in which a differentiable real function $f : \mathcal{R} \to \mathcal{R}$ with a dual argument $\alpha + \omega\beta$, where $\alpha, \beta \in \mathcal{R}$, can be expanded using a Taylor series. Because $\omega^2 = \omega^3 = \omega^4 = \cdots = 0$, the function reads

$$\begin{aligned} f(\alpha + \omega\beta) &= f(\alpha) + \omega f'(\alpha)\beta + \omega^2 f''(\alpha)\frac{\beta^2}{2!} + \cdots, \\ &= f(\alpha) + \omega f'(\alpha)\beta. \end{aligned} \tag{4.1}$$

© Springer Nature Switzerland AG 2020

E. Bayro-Corrochano, *Geometric Algebra Applications Vol. II*,
https://doi.org/10.1007/978-3-030-34978-3_4

A useful illustration of this expansion is the exponential function of a dual number,

$$e^{\alpha+\omega\beta} = e^{\alpha} + \omega e^{\alpha}\beta = e^{\alpha}(1 + \omega\beta). \tag{4.2}$$

In his seminal paper "Preliminary sketch of bi-quaternions" [3], Clifford introduced the use of dual numbers *the motors* or *bi-quaternions* to represent screw motion. Later, Study [4] used dual numbers to represent the relative position of two skew lines in space—that is, $\hat{\theta} = \theta + \omega d$, where $\hat{\theta}$ represents the dual angle, θ for the difference of the line orientation angles, and d for the distance between both lines.

The algebras of complex, double (hyperbolic), and dual numbers are isomorphic to the *center* of certain geometric algebras. For these algebras, we must choose the appropriate multivector basis, so that the unit pseudoscalar squares to 1 for the case of double numbers, to -1 for complex numbers, and to 0 for dual numbers. Note that the pseudoscalar for these numbers maintains its geometric interpretation as a unit hypervolume, and that, as is the case with ω, they are commutative with either vectors or bivectors, depending only upon the type of the geometric algebra used.

In Sect. 4.2, we will consider some examples of composed numbers in geometric algebra: complex numbers in the space $G_{0,1,0}$, double numbers in the space $G_{1,0,0}$, and dual complex numbers in the space $G_{1,0,1}$. We shall describe complex and dual numbers for 2D, 3D, and 4D spaces in some detail. The dual numbers will be used later for the modeling of points, lines, and planes, as well as for the modeling of motion.

4.2 2D Geometric Algebras of the Plane

In this section, we want to illustrate the application of different 2D geometric algebras for the modeling of group transformations on the plane. In doing so, we can also clearly see the geometric interpretation and the use of complex, double, and dual numbers for the cases of rotation, affine, and Lorentz transformations, respectively [2, 5]. We find these transformations in various tasks of image processing. For the modeling of the 2D space, we choose a geometric algebra which has $2^2 = 4$ elements, given by

$$\underbrace{1}_{scalar}, \quad \underbrace{e_1, e_2}_{vectors}, \quad \underbrace{e_1 e_2}_{bivector} \equiv I. \tag{4.3}$$

The highest grade element for the 2D space, called the unit pseudoscalar $I \equiv e_1 e_2$, is a bivector. According to the used vector basis, the signature of the geometric algebra will change, yielding complex, double, or dual numbers. Each of these cases is illustrated below.

In the geometric algebra $G_{2,0,0}$, where $I = e_1 e_2$ with $I^2 = -1$, we want to represent the rotation of the points (x, y) of the *Euclidean plane*. Here, a rotation of

the point $z = xe_1 + ye_2 = r(\cos(\alpha)e_1 + \sin(\alpha)e_2) \in G_{2,0,0}$ can be computed as the geometric product of the vector and the complex number $e^{I\frac{\theta}{2}} = \cos\frac{\theta}{2} + e_1e_2\sin\frac{\theta}{2} = (\cos\frac{\theta}{2} + I\sin\frac{\theta}{2}) \in G^+_{2,0,0}$, or $\in$ Spin(2) (spin group), as follows:

$$\begin{aligned} z' &= e^{-I\frac{\theta}{2}} z e^{I\frac{\theta}{2}}, \\ &= e^{-I\frac{\theta}{2}} r(\cos(\alpha)e_1 + \sin(\alpha)e_2) e^{I\frac{\theta}{2}}, \\ &= \left(\cos\frac{\theta}{2} + I\sin\frac{\theta}{2}\right)^{-1} r(\cos(\alpha)e_1 + \sin(\alpha)e_2)\left(\cos\frac{\theta}{2} + I\sin\frac{\theta}{2}\right), \\ &= r(\cos(\alpha+\theta)e_1 + \sin(\alpha+\theta)e_2). \end{aligned} \tag{4.4}$$

Figure 4.1b illustrates that each point of the 2D image of the die is rotated by θ. Note that this particular form for representing rotation, $e^{I\frac{\theta}{2}} = (\cos\frac{\theta}{2} + I\sin\frac{\theta}{2})$, can be generalized to higher dimensions (see the algebra of rotors in 3D space in the next section).

Let us now represent the points as dual numbers in the geometric algebra $\mathcal{G}_{1,0,1}$, where $I^2 = 0$. A 2D point can be represented in $G_{1,0,1}$ as $z = xe_1 + ye_2 = x(e_1 + se_2)$, where $s = \frac{y}{x}$ is the slope. The shear transformation of this point can be computed by applying a unit shear dual number $e^{I\frac{\tau}{2}} = (1 + I\frac{\tau}{2}) \in \mathcal{G}_{1,0,1}$ as follows:

$$\begin{aligned} z' &= e^{-I\frac{\tau}{2}} z e^{I\frac{\tau}{2}} = \left(1 - I\frac{\tau}{2}\right)(x(e_1 + se_2))\left(1 + I\frac{\tau}{2}\right), \\ &= x(e_1 + (s+\tau)e_2). \end{aligned} \tag{4.5}$$

Note that the overall effect of this transformation is to shear the plane, where the points (x, y) lie parallel to the e_2-axis through the shear τ with a shear angle of $\tan^{-1}\tau$. Figure 4.1c depicts the effect of the shear transformation acting on the 2D image of the die.

By using the representation of the double number in $G_{1,1,0}$, where $I^2 = 1$, we can implement the Lorentz transformation of the points. This transformation is commonly used in space–time algebra for special relativity computations, and it has been suggested for use in psychophysics as well [6, 7]. In this context, a 2D point is associated with a double number $z = te_1 + xe_2 = \rho(\cosh(\alpha)e_1 + \sinh(\alpha)e_2) \in G_{1,1,0}$. The lines $|t| = |x|$ divide the plane into two quadrants with $|t| > |x|$ and two quadrants with $|t| < |x|$. If we apply a 2D unit displacement vector $e^{I\frac{\beta}{2}} = a + Ib = (\cosh(\beta) + I\sinh(\beta)) \in G_{1,1,0}$ from one of the quadrants, $|t| > |x|$, to an arbitrary point $z = t + Ix$, we get

$$\begin{aligned} z' &= e^{-I\frac{\beta}{2}} z e^{I\frac{\theta}{2}}, \\ &= e^{-I\frac{\beta}{2}} \rho(\cosh(\alpha)e_1 + \sinh(\alpha)e_2) e^{I\frac{\theta}{2}}, \\ &= \left(\cosh\frac{\beta}{2} + I\sinh\frac{\beta}{2}\right)^{-1} (\rho(\cosh\alpha e_1 + \sinh\alpha e_2))\left(\cosh\frac{\beta}{2} + I\sinh\frac{\beta}{2}\right), \\ &= \rho(\cosh(\alpha+\beta)e_1 + \sinh(\alpha+\beta)e_2). \end{aligned} \tag{4.6}$$

The point is displaced along a particular hyperbolic path through the interval $\rho\beta$ in $|t| < |x|$. Figure 4.1d illustrates the effect of the Lorentz transformation acting on the 2D image of the die.

4.3 3D Geometric Algebra for the Euclidean 3D Space

For the case of embedding Euclidean 3D space, we choose the geometric algebra $G_{3,0,0}$, which has $2^3 = 8$ elements given by

$$\underbrace{1}_{scalar}, \quad \underbrace{\{e_1, e_2, e_3\}}_{vectors}, \quad \underbrace{\{e_2e_3, e_3e_1, e_1e_2\}}_{bivectors}, \quad \underbrace{\{e_1e_2e_3\} \equiv I}_{trivector}. \tag{4.7}$$

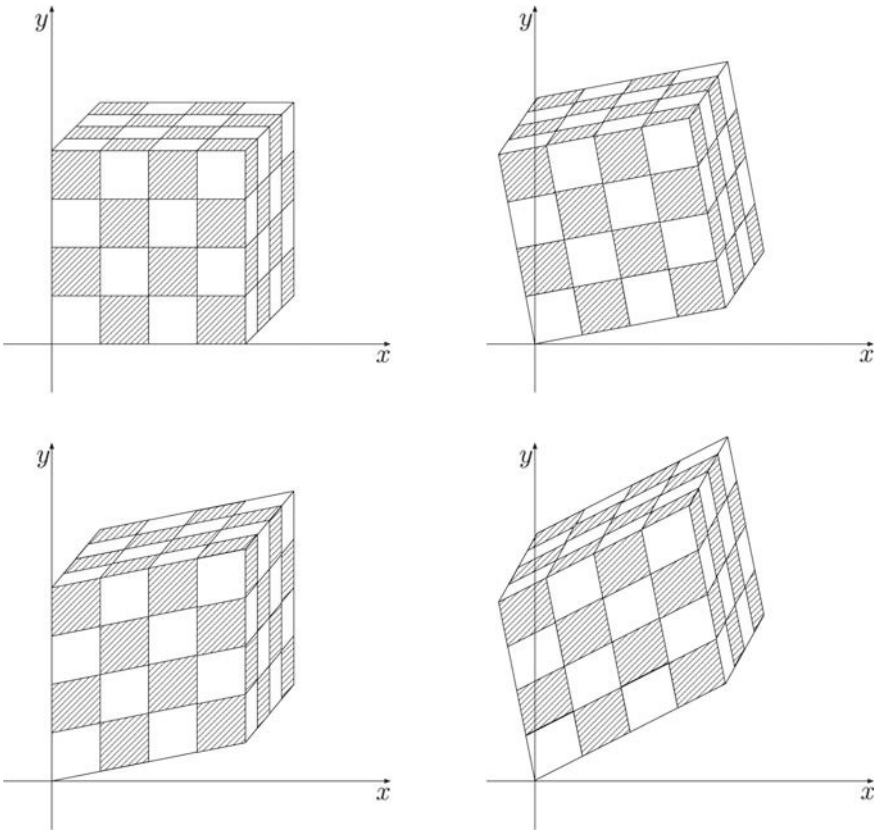


Fig. 4.1 Effects of 2D transformations: **a** original cube, **b** cube after rotation, **c** after shear transformation, and **d** after Lorentz transformation

The highest grade algebraic element for the 3D space is a trivector called a unit pseudoscalar $I \equiv e_1e_2e_3$, which squares to -1 and which commutes with the scalars and bivectors in the 3D space. In the algebra of three-dimensional space, we can construct a trivector $\boldsymbol{a} \wedge \boldsymbol{b} \wedge \boldsymbol{c} = \lambda I$, where the vectors $\boldsymbol{a}$, $\boldsymbol{b}$, and $\boldsymbol{c}$ are in general position and $\lambda \in \mathcal{R}$. Note that no 4-vectors exist since there is no possibility of sweeping the volume element $\boldsymbol{a} \wedge \boldsymbol{b} \wedge \boldsymbol{c}$ over a fourth dimension.

Multiplication of the three basis vectors e_1, e_2, and e_3 by I results in the three basis bivectors $e_2e_3 = Ie_1, e_3e_1 = Ie_2$, and $e_1e_2 = Ie_3$. These simple bivectors rotate vectors in their own plane by 90°, for example, $(e_1e_2)e_2 = e_1$, $(e_2e_3)e_2 = -e_3$, etc. Identifying the unit vectors $\boldsymbol{i}, \boldsymbol{j}, \boldsymbol{k}$ of quaternion algebra with $Ie_1, -I\sigma_2, I\sigma_3$ allows us to write the famous Hamilton relations $\boldsymbol{i}^2 = \boldsymbol{j}^2 = \boldsymbol{k}^2 = \boldsymbol{ijk} = -1$. Since the $\boldsymbol{i}, \boldsymbol{j}, \boldsymbol{k}$ are really bivectors, it comes as no surprise that they represent 90° rotations in orthogonal directions and provide a system well suited for the representation of general 3D rotations (see Fig. 2.2c). Rotors are isomorphic with quaternions. The quaternion and rotor follow the left-hand and the right-hand rotation rule, respectively.

4.3.1 The Algebra of Rotors

In geometric algebra a *rotor* (short name for rotator), $\boldsymbol{R}$, is an even-grade element of the Euclidean algebra of 3D-space. If $Q = \{r_0, r_1, r_2, r_3\} \in G_{3,0,0}$ represents a unit quaternion, then the rotor which performs the same rotation is simply given by

$$\boldsymbol{R} = \underbrace{r_0}_{scalar} + \underbrace{r_1(Ie_1) - r_2(Ie_2) + r_3(Ie_3)}_{bivectors}. \tag{4.8}$$

The rotor algebra $G^+_{3,0,0}$ is therefore a subset of the Euclidean geometric algebra of three-dimensional space.

Consider in $\mathcal{G}_{3,0,0}$ two nonparallel vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ which are referred to the same origin. In general, a rotation operation of a vector $\boldsymbol{a}$ toward the vector $\boldsymbol{b}$ can be performed by two *reflections*, respective to the unit vector axes $\boldsymbol{n}$ and $\boldsymbol{m}$ (see Fig. 4.2). The components of the first reflection are

$$\boldsymbol{a}_\parallel = |\boldsymbol{a}| \cos(\alpha) \frac{\boldsymbol{n}}{|\boldsymbol{n}|} = |\boldsymbol{a}||\boldsymbol{n}| \cos(\alpha) \frac{\boldsymbol{n}}{|\boldsymbol{n}|^2} = (\boldsymbol{a} \cdot \boldsymbol{n})\boldsymbol{n}^{-1}, \tag{4.9}$$

$$\begin{aligned} \boldsymbol{a}_\perp &= \boldsymbol{a} - \boldsymbol{a}_\parallel = \boldsymbol{a} - (\boldsymbol{a} \cdot \boldsymbol{n})\boldsymbol{n}^{-1} = (\boldsymbol{an} - \boldsymbol{a} \cdot \boldsymbol{n})\boldsymbol{n}^{-1}, \\ &= (\boldsymbol{a} \wedge \boldsymbol{n})\boldsymbol{n}^{-1}, \end{aligned} \tag{4.10}$$

so the vector $\boldsymbol{a}$ after the first reflection becomes

$$\begin{aligned} \boldsymbol{a}' &= \boldsymbol{a}_\parallel - \boldsymbol{a}_\perp = (\boldsymbol{a} \cdot \boldsymbol{n})\boldsymbol{n}^{-1} - (\boldsymbol{a} \wedge \boldsymbol{n})\boldsymbol{n}^{-1} = (\boldsymbol{a} \cdot \boldsymbol{n} - \boldsymbol{a} \wedge \boldsymbol{n})\boldsymbol{n}^{-1}, \\ &= (\boldsymbol{n} \cdot \boldsymbol{a} + \boldsymbol{n} \wedge \boldsymbol{a})\boldsymbol{n}^{-1} = \boldsymbol{nan}^{-1}. \end{aligned} \tag{4.11}$$

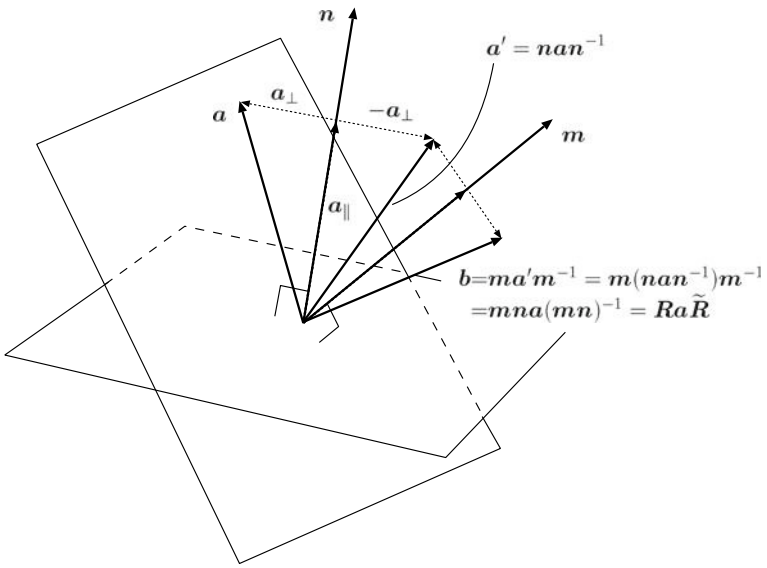


Fig. 4.2 Rotor in the 3D space formed by a pair of reflections

The second reflection respective to the axis unit $\boldsymbol{m}$ completes the vector rotation of $\boldsymbol{a}$ toward $\boldsymbol{b}$, as follows:

$$\begin{aligned} \boldsymbol{b} &= \boldsymbol{m}(\boldsymbol{a}')\boldsymbol{m}^{-1} = \boldsymbol{m}(\boldsymbol{nan}^{-1})\boldsymbol{m}^{-1} = \boldsymbol{mnan}^{-1}\boldsymbol{m}^{-1} = (\boldsymbol{mn})\boldsymbol{a}(\boldsymbol{mn})^{-1}, \\ &= \boldsymbol{RaR}^{-1} = \boldsymbol{Ra}\tilde{\boldsymbol{R}}. \end{aligned} \tag{4.12}$$

The rotor $\boldsymbol{R}$ composed of these two reflections performs a rotation that is two times greater than the angle between $\boldsymbol{m}$ and $\boldsymbol{n}$. Now, if we consider successive reflections of the vector $\boldsymbol{a}$ with respect to j planes, we get the following resultant transformation:

$$\begin{aligned} \boldsymbol{b} &= (\boldsymbol{m}_j \ldots \boldsymbol{m}_2\boldsymbol{m}_1)\boldsymbol{a}(\boldsymbol{m}_j \ldots \boldsymbol{m}_2\boldsymbol{m}_1)^{-1}, \\ &= (\boldsymbol{m}_j\boldsymbol{R}_{(j-1)(j-2)} \ldots \boldsymbol{R}_{32}\boldsymbol{R}_{21})\boldsymbol{a}(\boldsymbol{m}_j\boldsymbol{R}_{(j-1)(j-2)} \ldots \boldsymbol{R}_{32}\boldsymbol{R}_{21})^{-1}, \\ &= \boldsymbol{m}_j\boldsymbol{R}_{1(j-1)}\boldsymbol{a}\boldsymbol{R}_{(j-1)1}\boldsymbol{m}_j, \qquad \texttt{for j odd} \\ &= \boldsymbol{R}_{j1}\boldsymbol{a}\boldsymbol{R}_{1j}, \qquad \texttt{for j even.} \end{aligned} \tag{4.13}$$

Figure 4.3 shows the case for three reflections or $j = 3$.

According to Eqs. (2.92)–(2.97), the reversion and magnitude of a rotor $\boldsymbol{R}$ are, respectively, given by

$$\begin{aligned} \tilde{\boldsymbol{R}} &= r_0 - r_1e_2e_3 - r_2e_3e_1 - r_3e_1e_2 = r_0 - \boldsymbol{r}, \\ ||\boldsymbol{R}||^2 &= \boldsymbol{R}\tilde{\boldsymbol{R}}. \end{aligned} \tag{4.14}$$

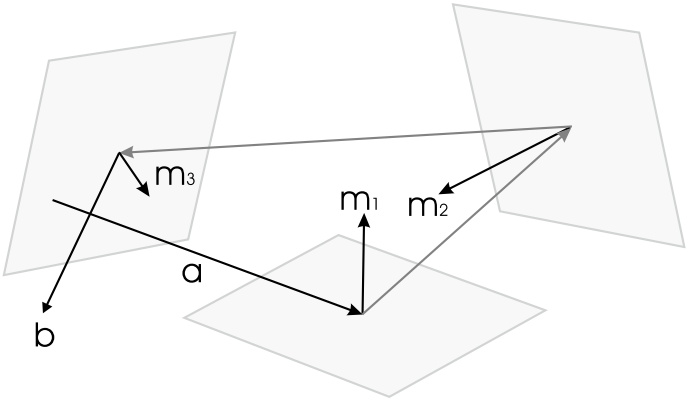


Fig. 4.3 Trajectory of successive reflections

This implies that the unique multiplicative inverse of $\boldsymbol{R}$ is given by

$$\boldsymbol{R}^{-1} = \tilde{\boldsymbol{R}} \parallel \boldsymbol{R} \parallel^{-2} . \tag{4.15}$$

If a rotor $\boldsymbol{R}$ satisfies the equation

$$\boldsymbol{R}\tilde{\boldsymbol{R}} = \parallel \boldsymbol{R} \parallel^2 = r_0^2 - \boldsymbol{r} \cdot \boldsymbol{r} = 1, \tag{4.16}$$

then we say that this rotor is a unit rotor and its multiplicative inverse is simply $\boldsymbol{R}^{-1} = \tilde{\boldsymbol{R}}$, as denoted previously in Eq. (4.12).

Equation (4.12) shows that the unit rotor corresponds to the geometric product of two unit vectors,

$$\boldsymbol{R} = \boldsymbol{mn} = \boldsymbol{m} \cdot \boldsymbol{n} + \boldsymbol{m} \wedge \boldsymbol{n}. \tag{4.17}$$

The components of Eq. (4.17) correspond to the scalar and bivector terms of an equivalent quaternion in $G_{3,0,0}$, and thus $\boldsymbol{R} \in G^{+}_{3,0,0}$. This even subalgebra corresponds to the algebra of rotors.

Considering the scalar and the bivector terms of the rotor of Eq. (4.17), we can further write the Euler representation of a 3D rotation with angle θ in the left-hand sense, as follows:

$$\begin{aligned} \boldsymbol{R} &= r_0 + \boldsymbol{r} = r_0 + r_1 e_2 e_3 + r_2 e_3 e_1 + r_3 e_1 e_2, \\ &= a_c + a_s \bar{\boldsymbol{r}}_n = \cos\left(\frac{\theta}{2}\right) + \sin\left(\frac{\theta}{2}\right) \bar{\boldsymbol{r}}_n, \\ &= e^{\frac{\theta}{2} \bar{\boldsymbol{r}}_n}, \end{aligned} \tag{4.18}$$

where $\bar{\boldsymbol{r}}_n$ is the unitary rotation-axis vector spanned by the bivector basis e_2e_3, e_3e_1, and e_1e_2, and the scalars a_c and $a_s \in \mathcal{R}$. The polar representation of a rotor given in Eq. (4.18) is possible, because the rotor as a Lie group can be expressed in terms of the Lie algebra of bivectors: The orbits on the Lie group manifold describe the evolution of the actions of rotors. The bivector $\bar{\boldsymbol{r}}_n$ corresponds to the Lie operator tangent to an orbit or geodesic.

The transformation of a rotor $\boldsymbol{p} \mapsto \mathbf{R}\boldsymbol{p}\tilde{\mathbf{R}} = \boldsymbol{p}'$ is a very general way of handling rotations which works for multivectors of any grade and in spaces of any dimension. Rotors combine in a straightforward manner—i.e., a rotor $\boldsymbol{R}_1$ followed by a rotor $\boldsymbol{R}_2$ is equivalent to a total rotor

$$\boldsymbol{R} = \boldsymbol{R}_2\boldsymbol{R}_1. \tag{4.19}$$

The composition of 3D rotations in different planes is described in Eqs. (4.13) and (4.19), the latter can be visualized geometrically in Fig. 4.4, where the half-angle of each rotor is depicted as a directed arc vector called $\boldsymbol{\theta}_i$ confined to a great circle on the unit sphere. The product of rotors $\boldsymbol{R}_1$ and $\boldsymbol{R}_2$ is depicted in Fig. 4.4 by connecting the corresponding arcs at the point $\boldsymbol{z}$ where the two great circles of $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$ intersect. As a result $\boldsymbol{x} = \boldsymbol{z}\boldsymbol{\theta}_1$ and $\boldsymbol{y} = \boldsymbol{\theta}_2\boldsymbol{z}$, thus the half-angle of the rotor can be computed depending of a common arc vector: $\boldsymbol{\theta}_1 = \boldsymbol{z}\boldsymbol{x}$ and $\boldsymbol{\theta}_2 = \boldsymbol{y}\boldsymbol{z}$. Combining these results, we get $\boldsymbol{\theta}_3 = \boldsymbol{\theta}_2\boldsymbol{\theta}_1 = (\boldsymbol{y}\boldsymbol{z})(\boldsymbol{z}\boldsymbol{x}) = \boldsymbol{y}\boldsymbol{x}$.

A rotor is isomorph with a quaternion. As a result, we can embed quaternions in the more comprehensive mathematical system offered by geometric algebra. Different as in quaternion theory, in geometric algebra the quaternions or rotors

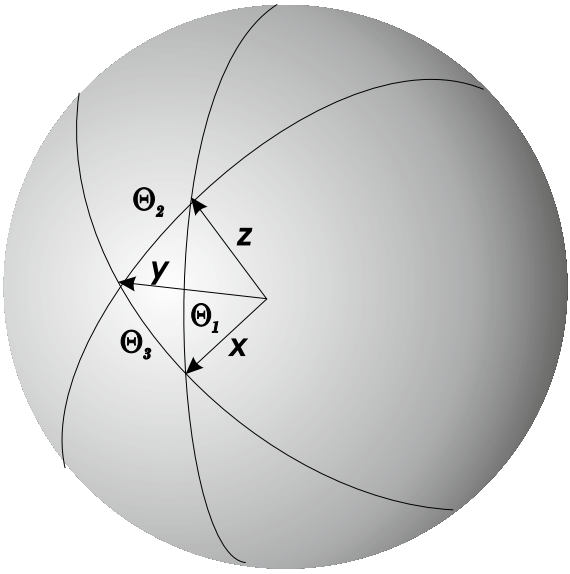


Fig. 4.4 Rotors represented on the sphere

have a clear geometric interpretation due to the representation in space of the rotations as described above by using reflections with respect to planes. Section 4.4 is devoted to quaternion algebra including more details useful for applications in image processing.

4.3.2 Orthogonal Rotors

For the rotation of a vector $\boldsymbol{p}$ in the right-hand sense, we simply adopt the rotor with the minus sign to agree with the standard right-hand rule for the direction of the rotation:

$$\begin{aligned} \boldsymbol{R} &= e^{-\frac{\bar{\boldsymbol{r}}_n\theta}{2}} = e^{-\frac{\theta}{2}\bar{\boldsymbol{r}}_n}, \\ &= \cos(\theta/2) - \sin(\theta/2)\bar{\boldsymbol{r}}_n. \end{aligned} \tag{4.20}$$

This rotation operation is depicted in Fig. 4.5. The rotated vector $\boldsymbol{p}'$ is given by

$$\boldsymbol{p}' = \boldsymbol{R}\boldsymbol{p}\tilde{\boldsymbol{R}} = \Big(\cos(\theta/2) - \sin(\theta/2)\bar{\boldsymbol{r}}_n\Big)\boldsymbol{p}\Big(\cos(\theta/2) + \sin(\theta/2)\bar{\boldsymbol{r}}_n\Big). \tag{4.21}$$

Since the rotation path from $\boldsymbol{p}$ and $\boldsymbol{p}'$ is not necessarily unique, neither is the rotor $\boldsymbol{R}$ unique. The shortest path determined by the endpoints $\boldsymbol{p}$ and $\boldsymbol{p}'$ lies on a great circle of a sphere with radius $\| \boldsymbol{p} \|$, and this is called *orthogonal rotation*. The rotor

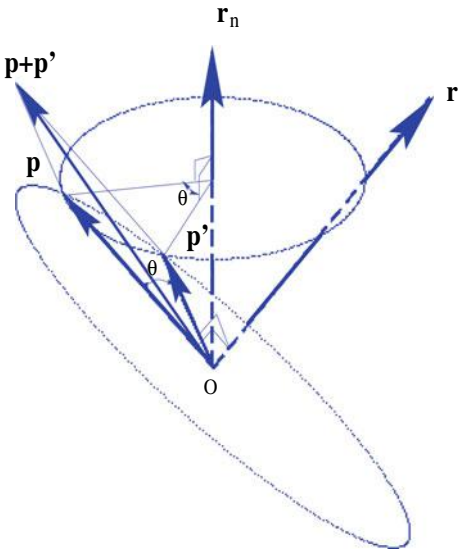


Fig. 4.5 Geometric interpretation of rotation

itself is called *orthogonal rotor* $\boldsymbol{R}_\perp$, and it can be calculated using the unit vectors $\frac{\boldsymbol{p}'+\boldsymbol{p}}{\|\boldsymbol{p}'+\boldsymbol{p}\|}$ and $\frac{\boldsymbol{p}}{\|\boldsymbol{p}\|}$, as follows:

$$\begin{aligned}\boldsymbol{R}_\perp &= -\frac{(\boldsymbol{p}'+\boldsymbol{p})\boldsymbol{p}}{\|\,\boldsymbol{p}'+\boldsymbol{p}\,\|\cdot\|\,\boldsymbol{p}\,\|} = -\frac{(\boldsymbol{p}'+\boldsymbol{p})\cdot\boldsymbol{p}+(\boldsymbol{p}'+\boldsymbol{p})\wedge\boldsymbol{p}}{\|\,\boldsymbol{p}'+\boldsymbol{p}\,\|\cdot\|\,\boldsymbol{p}\,\|},\\ &= -\frac{(\boldsymbol{p}'+\boldsymbol{p})\cdot\boldsymbol{p}}{\|\,\boldsymbol{p}'+\boldsymbol{p}\,\|\cdot\|\,\boldsymbol{p}\,\|} - \frac{-\boldsymbol{p}\wedge(\boldsymbol{p}'+\boldsymbol{p})}{\|\,\boldsymbol{p}'+\boldsymbol{p}\,\|\cdot\|\,\boldsymbol{p}\,\|},\\ &= \boldsymbol{r}_{\perp 0}+\boldsymbol{r}_\perp = \cos(\theta_\perp/2) - \sin(\theta_\perp/2)\boldsymbol{r}_{n,\perp},\end{aligned} \tag{4.22}$$

where the rotation-axis bivector $\boldsymbol{r}_\perp$ and the unit rotation-axis bivector $\boldsymbol{r}_{n,\perp}$ are perpendicular to both $\boldsymbol{p}$ and $\boldsymbol{p}'$, and the angle $\theta_\perp/2$ is the angle between the vectors $\boldsymbol{p}$ and $\boldsymbol{p}'+\boldsymbol{p}$. Rotors are isomorphic with quaternions. In signal analysis, quaternions have been used quite often in an operational sense. In contrast, rotors have a clear geometric interpretation, and they are used for geometric operations.

4.3.3 Recovering a Rotor

Using previous results, we will derive a procedure to recover a rotor based on two observer measurements. Assume that we have two sets of vectors in 3D $\{e_k\}$ and $\{f_k\}$ which are not necessarily orthonormal and that they are related by a rotation

$$f_k = \boldsymbol{R}e_k\tilde{\boldsymbol{R}}. \tag{4.23}$$

Since the $\boldsymbol{R}$ is unknown, we should derive a simple method to recover $\boldsymbol{R}$. A rotor can be written as

$$\begin{aligned}\boldsymbol{R} &= e^{-\frac{\boldsymbol{B}}{2}},\\ \tilde{\boldsymbol{R}} &= e^{\frac{\boldsymbol{B}}{2}} = \cos\left(\frac{|\boldsymbol{B}|}{2}\right)+\sin\left(\frac{|\boldsymbol{B}|}{2}\right)\frac{\boldsymbol{B}}{|\boldsymbol{B}|}.\end{aligned} \tag{4.24}$$

Using the equation

$$\begin{aligned}e_k\tilde{\boldsymbol{R}}e^k &= e_k\left[\cos\left(\frac{|\boldsymbol{B}|}{2}\right)+\sin\left(\frac{|\boldsymbol{B}|}{2}\right)\frac{\boldsymbol{B}}{|\boldsymbol{B}|}\right]e^k,\\ &= 3\cos\left(\frac{|\boldsymbol{B}|}{2}\right)-\sin\left(\frac{|\boldsymbol{B}|}{2}\right)\frac{\boldsymbol{B}}{|\boldsymbol{B}|},\\ &= 4\cos\left(\frac{|\boldsymbol{B}|}{2}\right)-\tilde{\boldsymbol{R}}.\end{aligned} \tag{4.25}$$

Reusing Eq. (4.23)

$$f_k e^k = \boldsymbol{R} e_k \tilde{\boldsymbol{R}} e^k = 4\cos\left(\frac{|\boldsymbol{B}|}{2}\right)\boldsymbol{R} - 1. \tag{4.26}$$

We can see that the unknown rotor is a scalar multiple of $1 + f_k e^k$; thus, one can establish the simple and useful formula

$$\boldsymbol{R} = \frac{1 + f_k e^k}{|1 + f_k e^k|} = \frac{\zeta}{\sqrt{\zeta\tilde{\zeta}}}, \tag{4.27}$$

where $\zeta = 1 + f_k e^k$. Using this formula, one can recover the rotor directly from the frame vectors.

4.4 Quaternion Algebra

The quaternion algebra $\mathbb{H}$ was invented by W. R. Hamilton in 1843 [8, 9], when he intended for almost 10 years to find an algebraic system which would do for the space $\mathbb{R}^3$ the same as complex numbers do for the space $\mathbb{R}^2$. Interesting enough, the current formalism of vector algebra was simply extracted out from the quaternion product of two vectors by Gibbs in 1901, namely, $\boldsymbol{ab} = -\boldsymbol{a}\cdot\boldsymbol{b} + \boldsymbol{a}\times\boldsymbol{b}$.

Hamilton tried to find a multiplications rule for triplets $\boldsymbol{a} = a_1\boldsymbol{i} + a_2\boldsymbol{j} + a_3\boldsymbol{k}$ and $\boldsymbol{b} = b_1\boldsymbol{i} + b_2\boldsymbol{j} + b_3\boldsymbol{k}$, so that $|\boldsymbol{ab}| = |\boldsymbol{a}||\boldsymbol{b}|$ correspond to a multiplicative product of the vectors $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^3$. However, according to a result of Legendre (1830) such bilinear product confined in $\mathbb{R}^3$ does not exist, i.e., no integer of the form $4^a(6b + 7)$ with $a \geq 0,\ b \geq 0$ can be obtained as a sum of three squares. Hamilton searched for a generalized complex number system in three dimensions, but no such associative hypercomplex numbers exist in three dimensions. One can see this fact easily by considering two imaginary units $\boldsymbol{i}$ and $\boldsymbol{j}$ such that $\boldsymbol{i}^2 = \boldsymbol{j}^2 = -1$ and furthermore that $1, \boldsymbol{i}, \boldsymbol{j}$ span $\mathbb{R}^3$. So the multiplication has to be of the form $\boldsymbol{ij} = \alpha + \boldsymbol{i}\beta + \boldsymbol{j}\gamma$ for $\alpha, \beta, \gamma \in \mathbb{R}$. Let us prove its consistency, on the one hand,

$$\boldsymbol{i}(\boldsymbol{ij}) = \alpha\boldsymbol{i} + -\beta + \gamma(\boldsymbol{ij}) = -\beta + \alpha\boldsymbol{i} + \gamma(\alpha + \boldsymbol{i}\beta + \boldsymbol{j}\gamma), \tag{4.28}$$

$$= -\beta + \alpha\gamma + (\alpha + \beta\gamma)\boldsymbol{i} + \gamma^2\boldsymbol{j}. \tag{4.29}$$

On the other hand by associativity

$$\boldsymbol{i}(\boldsymbol{ij}) = (\boldsymbol{ii})\boldsymbol{j} = \boldsymbol{i}^2\boldsymbol{j} = -\boldsymbol{j}, \tag{4.30}$$

which is a contradiction with the above equation, since $\gamma^2 \geq 0$ for any $\gamma \in \mathbb{R}$.

The Frobenius theorem proved by Ferdinand Georg Frobenius in 1877 characterizes the finite-dimensional associative algebras over the real numbers. It proves that, if A is a finite-dimensional division algebra over the real numbers R,

then one of the following cases is true: $A = \mathbb{R}$, $A = \mathbb{C}$ (complex numbers) and A is isomorphic to the quaternion algebra $\mathbb{H}$.

The key idea of Hamilton's discovery was to move to a four-dimensional space and to consider elements of the form $\boldsymbol{q} = s + x\boldsymbol{i} + y\boldsymbol{j} + z\boldsymbol{k}$ for $s, x, y, z \in \mathbb{R}$, where the orthogonal imaginary numbers $\boldsymbol{i}$, $\boldsymbol{j}$ and $\boldsymbol{k}$ obey the following multiplicative rules:

$$\boldsymbol{i}^2 = \boldsymbol{j}^2 = -1, \ \boldsymbol{k} = \boldsymbol{i}\boldsymbol{j} = -\boldsymbol{j}\boldsymbol{i} \ \rightarrow \ \boldsymbol{k}^2 = -1. \tag{4.31}$$

Hamilton named his four-component elements *quaternions*. Quaternions form a division ring and are denoted by H in honor of Hamilton. The quaternion algebra is non-associative. Arthur Cayley (1821–1895) was the first person, after Hamilton, to publish a paper on quaternions.

The conjugated of a quaternion is given by

$$\bar{\boldsymbol{q}} = s - x\boldsymbol{i} - y\boldsymbol{j} - z\boldsymbol{k}. \tag{4.32}$$

Arthur Cayley in 1855 discovered the quaternionic representation of four-dimensional rotations, namely,

$$\mathbb{R}^4 \rightarrow \mathbb{R}^4, \qquad \boldsymbol{q}_1 \rightarrow \boldsymbol{q}_2\boldsymbol{q}_1\bar{\boldsymbol{q}}_2, \tag{4.33}$$

where $\mathbb{R}^4$ stands for $\mathbb{H}$ and $\boldsymbol{q}_1, \boldsymbol{q}_1 \in \mathbb{H}$.

For the quaternion $\boldsymbol{q}$, we can compute its partial angles as

$$\arg_i(q) = s\tan 2(x, s), \ \arg_j(q) = s\tan 2(y, s), \ \arg_k(q) = s\tan 2(z, s), \tag{4.34}$$

and its partial modules and its projections on its imaginary axes as

$$\begin{aligned} &mod_i(q) = \sqrt{s^2 + x^2}, \ mod_j(s) = \sqrt{s^2 + y^2}, \\ &mod_k(q) = \sqrt{s^2 + z^2}, \\ &mod_i(q)\exp(i\arg_i(q)) = s + xi, \ \ mod_i(q)\exp(j\arg_j(q)) = s + yj, \\ &mod_k(q)\exp(k\arg_k(q)) = s + zk. \end{aligned} \tag{4.35}$$

In signal analysis, quaternions have been used quite often in an operational sense. In contrast, rotors were introduced for geometric operations. Next, we will provide some definitions for quaternions which will be useful for the analysis and processing of images.

In a similar way as the complex numbers which can be expressed in a polar representation, we can also represent a quaternion in a polar form. The polar representation of a quaternion $\boldsymbol{q} = r + x\boldsymbol{i} + y\boldsymbol{j} + z\boldsymbol{k} \in G^+_{3,0,0}$ is given when the quaternion, seen as a Lie group, is expressed in terms of the Lie algebra of bivectors:

$$\boldsymbol{q} = |\boldsymbol{q}|e^{\boldsymbol{e}_2\boldsymbol{e}_3\phi}e^{\boldsymbol{e}_1\boldsymbol{e}_2\psi}e^{\boldsymbol{e}_1\boldsymbol{e}_3\theta} = |\boldsymbol{q}|e^{\boldsymbol{i}\phi}e^{\boldsymbol{k}\psi}e^{\boldsymbol{j}\theta}, \tag{4.36}$$

in which $(\phi, \theta, \psi) \in [-\pi, \pi[\times[-\frac{\pi}{2}, \frac{\pi}{2}[\times, \pi[- \times [-\frac{\pi}{4}, \frac{\pi}{4}]$.

For a unit quaternion $\boldsymbol{q} = q_0 + q_x i + q_y j + q_z k$, $|\boldsymbol{q}| = 1$, its phase can be evaluated first by computing $\psi = -\frac{\arcsin(2(q_x q_y - q_0 q_z))}{2}$ and then by checking that it adheres to the following rules:

- If $\psi \in]-\frac{\pi}{4}, \frac{\pi}{4}[$, then $\phi = \frac{\arg(\boldsymbol{q}\mathcal{I}_j(\bar{\boldsymbol{q}}))}{2}$ and $\theta = \frac{\arg(\mathcal{I}_i(\bar{\boldsymbol{q}})\boldsymbol{q})}{2}$.
- If $\psi = \pm\frac{\pi}{4}$, then select either $\phi = 0$ and $\theta = \frac{\arg(\mathcal{I}_k(\bar{\boldsymbol{q}})\boldsymbol{q})}{2}$ or $\theta = 0$ and $\phi = \frac{\arg(\boldsymbol{q}\mathcal{I}_k(\bar{\boldsymbol{q}}))}{2}$.
- If $e^{\boldsymbol{i}\phi}e^{\boldsymbol{k}\psi}e^{\boldsymbol{j}\theta} = -\boldsymbol{q}$ and $\phi \geq 0$, then $\phi \rightarrow \phi - \pi$.
- If $e^{\boldsymbol{i}\phi}e^{\boldsymbol{k}\psi}e^{\boldsymbol{j}\theta} = -\boldsymbol{q}$ and $\phi < 0$, then $\phi \rightarrow \phi + \pi$.

The reader can find the details of the development of these rules in [10].

The concept of quaternionic Hermitian function is very useful for the computation of the inverse quaternionic Fourier transform using the quaternionic analytic signal. As an extension of the Hermitian function $f : \mathbb{R} \rightarrow \boldsymbol{I}\mathbb{C}$ with $f(x) = f^*(-x)$ for every $x \in \mathbb{R}$, we regard $f : \mathbb{R}^2 \rightarrow \mathbb{H}$ as a quaternionic Hermitian function if it fulfills the following nontrivial involution rules [11]:

$$\begin{aligned} f(-x, y) &= -jf(x, y)j = \mathcal{T}_j(f(x, y)), \\ f(x, -y) &= -if(x, y)i = \mathcal{T}_i(f(x, y)), \\ f(-x, -y) &= -if(-x, y)i = -i(-jf(x, y)j)i = (-i - j)f(x, y)(ji), \\ &= -kf(x, y)k = \mathcal{T}_k(f(x, y)). \end{aligned} \tag{4.37}$$

4.5 4D Geometric Algebra for 3D Kinematics

Usually, problems of robotics are treated in algebraic systems of 2D and 3D space. In the case of 3D rigid motion, or Euclidean transformation, we are confronted with a nonlinear mapping; however, if we employ homogeneous coordinates in 4D geometric algebra, we can linearize the rigid motion in 3D Euclidean space. That is why we choose three basis vectors which square to one and a fourth vector which squares to zero—to provide dual copies of the multivectors of the 3D space. In other words, we extend the Euclidean geometric algebra $G_{3,0,0}$ to the special or degenerated geometric algebra $G_{3,0,1}$, which is spanned via the following basis:

$$\underbrace{1}_{scalar}, \quad \underbrace{e_k}_{4\ vectors}, \quad \underbrace{e_2e_3, e_3e_1, e_1e_2, e_4e_1, e_4e_2, e_4e_3}_{6\ \ bivectors} \quad \underbrace{Ie_k}_{4\ \ pseudovectors}, \quad \underbrace{I}_{unit\ \ pseudoscalar} \tag{4.38}$$

where $e_4^2 = 0$, $e_k^2 = +1$ for $k = 1, 2, 3$. The unit pseudoscalar is $I = e_1e_2e_3e_4$, with

$$I^2 = (e_1e_2e_3e_4)(e_1e_2e_3e_4) = -(e_3e_4)(e_3e_4) = 0. \tag{4.39}$$

The motor algebra $G^+_{3,0,1}$, which is the even subalgebra of $G_{3,0,1}$, can be utilized to obtain linear 4D models of the 3D motion of points, lines, and planes.

4.5.1 Motor Algebra

The word *motor* is an abbreviation of "moment and vector." Clifford introduced motors with the name bi-quaternions [3]. Motors are isomorphic to dual quaternions, with the necessary condition $I^2 = 0$. They can be found in the special 4D even subalgebra of $G_{3,0,1}$ introduced in Sect. 4.5. This even subalgebra is denominated by $G^+_{3,0,1}$ and is only spanned via a bivector basis, as follows:

$$\underbrace{1}_{scalar}, \quad \underbrace{e_2e_3, e_3e_1, e_1e_2, e_4e_1, e_4e_2, e_4e_3}_{6 \ \ bivectors}, \quad \underbrace{I}_{unit \ pseudoscalar}. \tag{4.40}$$

This kind of basis structure also allows us to represent spinors, which are composed of scalar and bivector terms. Motors, then, are also spinors, and as such, they represent a special kind of rotor. Because a Euclidean transformation includes both rotation and translation, we will show in the following subsection a spinor representation for both transformations in the definition of motors. But we must first show the relationship between motors and screw motion theory.

Note that the bivector terms of the basis correspond to the same basis for spanning 3D lines. Note also that the dual of a scalar is the pseudoscalar P and that the duals of the first three basis bivectors are actually the next three bivectors, that is, $(e_2e_3^*) = Ie_2e_3 = e_4e_1$.

We said in Sect. 4.3.1 that a rotor relates two vectors in 3D space. According to Clifford [3], a motor operation is necessary to convert the rotation axis of a rotor into the rotation axis of a second rotor. Each rotor can be geometrically represented as a rotation plane with the rotation axis normal to this plane. Figure 4.6a depicts a motor action in detail. Note that the involved rotor axes are represented as line axes. In the figure, we first orient one axis parallel to the other by applying the rotor $\boldsymbol{R}_s$. Then, we slide the rotated axis a distance d along the connecting axis, so that it ends up overlapping the axis of the second rotor. Altogether, this operation can be described as forming a *twist* about a screw with line axis $\boldsymbol{l}$, whose pitch relationship *pitch* equals $\frac{d}{\theta}$ for $\theta \neq 0$. A motor, then, is specified only by its direction and the position of the *screw-axis line*, twist angular magnitude, and pitch. Figure 4.6b shows an action of a motor on a real object. In this case, the motor relates the rotation-axis line of the initial position of the object to the rotation-axis line of its final position. Note that in both figures the angle and sliding distance indicate how rigid displacement takes place around and along a screw-axis line $\boldsymbol{l}$, respectively. A *degenerated motor* can only rotate and not slide along the line $\boldsymbol{l}$ as shown in Fig. 4.6c. In this case, therefore, the two axes are coplanar.

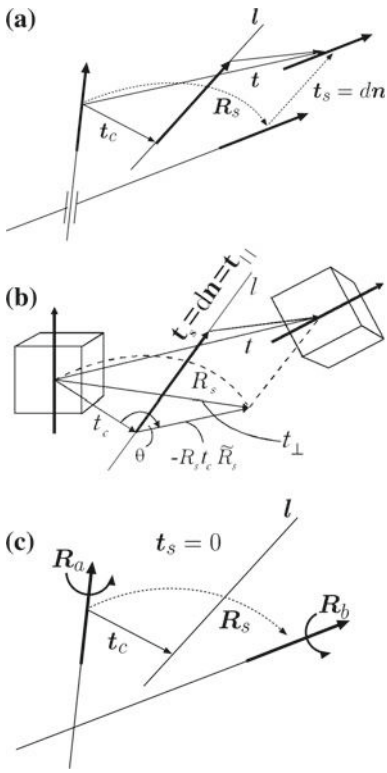


Fig. 4.6 Screw motion about the line axis $\boldsymbol{l}$ ($\boldsymbol{t}_s$: longitudinal displacement by d and $\boldsymbol{R}_s$: rotation angle θ): **a** motor relating two axis lines, **b** motor applied to an object, **c** degenerated motor relating two coplanar rotors (*Note* indicated 3D vectors are represented as bivectors in text.)

4.5.2 Motors, Rotors, and Translators in $G^+_{3,0,1}$

Since a rigid motion consists of the rotation and translation transformations, it should be possible to split a motor multiplicatively in terms of these two spinor transformations, which we will call a rotor and a *translator*. In the following discussion, we will denote all bivector components of a spinor by bold lowercase letters. Let us now express this procedure algebraically. First of all, let us consider a simple rotor in its *Euler representation* for a rotation with an angle θ,

$$\begin{aligned} \boldsymbol{R} &= a_0 + a_1 e_2 e_3 + a_2 e_3 e_1 + a_3 e_1 e_2, \\ &= a_0 + \boldsymbol{a}, \\ &= \cos\left(\frac{\theta}{2}\right) + \sin\left(\frac{\theta}{2}\right)\boldsymbol{n}, \\ &= a_c + a_s \boldsymbol{n}, \end{aligned} \tag{4.41}$$

where $\boldsymbol{n}$ is the unit 3D bivector of the rotation axis spanned by the bivector basis e_2e_3, e_3e_1, e_1e_2, and $a_c, a_s \in \mathcal{R}$. Now, dealing with the rotor of a screw motion, the

rotation-axis vector should be represented as a screw-axis line. For that, we must relate the rotation axis to a reference coordinate system at the distance $\boldsymbol{t}_c$. A 3D translation in motor algebra is represented by a spinor $\boldsymbol{T}_c$ called a translator. If we apply a translator from the left to rotor $\boldsymbol{R}$, and then apply the translator's conjugate from the right, we get a modified rotor,

$$\begin{aligned}
\boldsymbol{R}_s &= \boldsymbol{T}_c \boldsymbol{R} \widetilde{\boldsymbol{T}}_c, \\
&= \left(1 + I\frac{\boldsymbol{t}_c}{2}\right)(a_0 + \boldsymbol{a})\left(1 - I\frac{\boldsymbol{t}_c}{2}\right), \\
&= a_0 + \boldsymbol{a} + Ia_0\frac{\boldsymbol{t}_c}{2} + I\frac{\boldsymbol{t}_c}{2}\boldsymbol{a} - Ia_0\frac{\boldsymbol{t}_c}{2} - I\boldsymbol{a}\frac{\boldsymbol{t}_c}{2}, \\
&= a_0 + \boldsymbol{a} + I\left(\frac{\boldsymbol{t}_c}{2}\boldsymbol{a} - \boldsymbol{a}\frac{\boldsymbol{t}_c}{2}\right), \\
&= a_0 + \boldsymbol{a} + I(\boldsymbol{a} \times \boldsymbol{t}_c).
\end{aligned} \tag{4.42}$$

Here, $\boldsymbol{t}_c$ is the 3D vector of translation spanned by the bivector basis e_2e_3, e_3e_1, and e_1e_2. Then, expressing the last equation in Euler terms, we get the spinor representation,

$$\begin{aligned}
\boldsymbol{R}_s &= a_0 + a_s\boldsymbol{n} + Ia_s\boldsymbol{n} \wedge \boldsymbol{t}_c, \\
&= a_c + a_s(\boldsymbol{n} + I\boldsymbol{m}), \\
&= \cos\left(\frac{\theta}{2}\right) + \sin\left(\frac{\theta}{2}\right)(\boldsymbol{n} + I\boldsymbol{m}), \\
&= \cos\left(\frac{\theta}{2}\right) + \sin\left(\frac{\theta}{2}\right)\boldsymbol{l}.
\end{aligned} \tag{4.43}$$

This result is indeed interesting because the new rotor $\boldsymbol{R}_s$ can now be applied with respect to an axis line $\boldsymbol{l}$ expressed in dual terms of direction $\boldsymbol{n}$ and moment $\boldsymbol{m} = \boldsymbol{n} \wedge \boldsymbol{t}_c$. Now, to define the motor finally, let us slide the distance $\boldsymbol{t}_s = d\boldsymbol{n}$ along the rotation-axis line $\boldsymbol{l}$. Since a motor is applied from the left and conjugated from the right, we should use the half of $\boldsymbol{t}_s$ in the spinor expression of $\boldsymbol{T}_s 1$ when we define the motor:

$$\begin{aligned}
\boldsymbol{M} = \boldsymbol{T}_s\boldsymbol{R}_s &= \left(1 + I\frac{\boldsymbol{t}_s}{2}\right)(a_0 + \boldsymbol{a} + I\boldsymbol{a} \wedge \boldsymbol{t}_c), \\
&= \left(1 + I\frac{d\boldsymbol{n}}{2}\right)(a_c + a_s\boldsymbol{n} + Ia_s\boldsymbol{n} \wedge \boldsymbol{t}_c), \\
&= a_c + a_s\boldsymbol{n} + Ia_s\boldsymbol{n} \wedge \boldsymbol{t}_c + I\frac{d}{2}a_c\boldsymbol{n} - I\frac{d}{2}a_s\boldsymbol{n}\boldsymbol{n}, \\
&= \left(a_c - I\frac{d}{2}a_s\right) + \left(a_s + Ia_c\frac{d}{2}\right)(\boldsymbol{n} + I\boldsymbol{n} \wedge \boldsymbol{t}_c),
\end{aligned}$$

$$= \left(a_c - I a_s \frac{d}{2}\right) + \left(a_s + I a_c \frac{d}{2}\right) \boldsymbol{l}. \tag{4.44}$$

Note that this expression of the motor makes explicit the unit line bivector of the screw-axis line $\boldsymbol{l}$.

Now let us express a motor using Euler representation. By substituting the constants $a_c = \cos(\frac{\theta}{2})$ and $a_s = \sin(\frac{\theta}{2})$ in the motor Eq. (4.44) and using the property of Eq. (4.1), we get

$$\begin{aligned} \boldsymbol{M} = \boldsymbol{T}_s \boldsymbol{R}_s &= \left(\cos\left(\frac{\theta}{2}\right) - I \sin\left(\frac{\theta}{2}\right)\frac{d}{2}\right) + \left(\sin\left(\frac{\theta}{2}\right) + I \cos\left(\frac{\theta}{2}\right)\frac{d}{2}\right)\boldsymbol{l}, \\ &= \cos\left(\frac{\theta}{2} + I\frac{d}{2}\right) + \sin\left(\frac{\theta}{2} + I\frac{d}{2}\right)\boldsymbol{l}, \end{aligned} \tag{4.45}$$

which is a dual-number representation of the spinor. Now, let us analyze the resultant expressions,

$$\begin{aligned} \boldsymbol{R} &= \cos\left(\frac{\theta}{2}\right) + \sin\left(\frac{\theta}{2}\right)\boldsymbol{n}, \\ \boldsymbol{R}_s &= \cos\left(\frac{\theta}{2}\right) + \sin\left(\frac{\theta}{2}\right)\boldsymbol{l}, \\ \boldsymbol{M} &= \cos\left(\frac{\theta}{2} + I\frac{d}{2}\right) + \sin\left(\frac{\theta}{2} + I\frac{d}{2}\right)\boldsymbol{l}. \end{aligned} \tag{4.46}$$

We can see that the rotation axis $\boldsymbol{n}$ of the simple rotor $\boldsymbol{R}$ is changed to a rotation-axis line, so that $\boldsymbol{R}_s$ now rotates about an axis line. And in the motor expression, the information for sliding distance d is now made explicit in terms of dual arguments of the trigonometric functions. It is also interesting to note that the expression for the motor using dual angles simply extends the expression of $\boldsymbol{R}_s$.

If we expand the exponential function of the dual bivectors using a Taylor series, the result will follow the general expression $e^{\alpha + I\beta} = e^{\alpha} + I e^{\alpha}\beta = e^{\alpha}(1 + I\beta)$, which is a special case of Eq. (4.1). Once again, we obtain the motor expression as the spinor

$$e^{I\frac{\theta}{2} + I\frac{t_s}{2}} = \left(1 + I\frac{\boldsymbol{t}_s}{2}\right) e^{I\frac{\theta}{2}} = \boldsymbol{T}_s \boldsymbol{R}_s, \tag{4.47}$$

where $I\frac{t_s}{2} = I\frac{1}{2}(t_1\sigma_2\sigma_3 + t_2\sigma_3\sigma_1 + t_3\sigma_1\sigma_2) = \frac{1}{2}(t_1\sigma_4\sigma_1 + t_2\sigma_4\sigma_2 + t_3\sigma_4\sigma_3)$.

If we want to express the motor using only rotors in a dual spinor representation, we proceed as follows:

$$\begin{aligned} \boldsymbol{M} = \boldsymbol{T}_s \boldsymbol{R}_s &= \left(1 + I\frac{\boldsymbol{t}_s}{2}\right)\mathbf{R}_s, \\ &= \boldsymbol{R}_s + I\frac{\boldsymbol{t}_s}{2}\boldsymbol{R}_s. \end{aligned} \tag{4.48}$$

Let us consider carefully the resultant dual part of the motor. This is the geometric product of the bivector $\boldsymbol{t}_s$ and the rotor $\boldsymbol{R}_s$. Since both are expressed in terms of the same bivector basis, their geometric product will be also expressed in this basis, which can be considered as a new rotor $\boldsymbol{R}'_s$. Thus, we can further write

$$\boldsymbol{M} = \boldsymbol{R}_s + I\frac{\boldsymbol{t}_s}{2}\boldsymbol{R}_s = \boldsymbol{R}_s + I\boldsymbol{R}'_s. \tag{4.49}$$

In this equation, the line axes of the rotors are skewed (see Fig. 4.6a). This means that they represent the general case of non-coplanar rotors. If the sliding distance $\boldsymbol{t}_s$ is zero, then the motor will degenerate to a rotor

$$\boldsymbol{M} = \boldsymbol{T}_s\boldsymbol{R}_s = \left(1 + I\frac{\boldsymbol{t}_s}{2}\right)\boldsymbol{R}_s = \left(1 + I\frac{0}{2}\right)\boldsymbol{R}_s = \boldsymbol{R}_s. \tag{4.50}$$

In this case, that is, when the two generating axis lines of the motor are coplanar, we get the so-called *degenerated motor* (see Fig. 4.6c).

Finally, the bivector $\boldsymbol{t}_s$ can be expressed in terms of the rotors using previous results

$$\boldsymbol{R}'_s\widetilde{\boldsymbol{R}}_s = \left(\frac{\boldsymbol{t}_s}{2}\boldsymbol{R}_s\right)\widetilde{\boldsymbol{R}}_s; \tag{4.51}$$

therefore,

$$\boldsymbol{t}_s = 2\boldsymbol{R}'_s\widetilde{\boldsymbol{R}}_s. \tag{4.52}$$

Figure 4.6 shows that the 3D vector $\boldsymbol{t}$, expressed in the bivector basis, is referred to the rotation axis of the rotor, and that $\boldsymbol{t}_s$ is a bivector along the motor-axis line. Thus, $\boldsymbol{t}$, considered here as a bivector, can be computed in terms of the bivectors $\boldsymbol{t}_c$ and $\boldsymbol{t}_s$, as follows:

$$\begin{aligned}
&\boldsymbol{t} = \boldsymbol{t}_\perp + \boldsymbol{t}_\parallel,\\
&\boldsymbol{t} = (\boldsymbol{t}_c - \boldsymbol{R}_s\boldsymbol{t}_c\widetilde{\boldsymbol{R}}_s) + (\boldsymbol{t}\cdot\boldsymbol{n})\boldsymbol{n} = (\boldsymbol{t}_c - \boldsymbol{R}_s\boldsymbol{t}_c\widetilde{\boldsymbol{R}}_s) + d\boldsymbol{n},\\
&\quad = \boldsymbol{t}_c - \boldsymbol{R}_s\boldsymbol{t}_c\widetilde{\boldsymbol{R}}_s + \boldsymbol{t}_s,\\
&\quad = \boldsymbol{t}_c - \boldsymbol{R}_s\boldsymbol{t}_c\widetilde{\boldsymbol{R}}_s + 2\boldsymbol{R}'_s\widetilde{\boldsymbol{R}}_s.
\end{aligned} \tag{4.53}$$

So far, we have analyzed the motor from a geometrical point of view. Next, we will look at the motor's relevant algebraic properties.

4.5.3 Properties of Motors

A general motor can be expressed as

$$\boldsymbol{M}_{\alpha} = \alpha \boldsymbol{M}, \tag{4.54}$$

where $\alpha \in \mathcal{R}$ and $\boldsymbol{M}$ is a unit motor, as explained in the previous sections. In this section, we will employ unit motors. The norm of a motor $\boldsymbol{M}$ is defined as follows:

$$\begin{aligned} |\boldsymbol{M}| = \boldsymbol{M}\widetilde{\boldsymbol{M}} = \boldsymbol{T}_s \boldsymbol{R}_s \widetilde{\boldsymbol{R}}_s \widetilde{\boldsymbol{T}}_s = \left(1 + I\frac{\boldsymbol{t}_s}{2}\right) \boldsymbol{R}_s \tilde{\boldsymbol{R}}_s \left(1 - I\frac{\boldsymbol{t}_s}{2}\right), \\ = 1 + I\frac{\boldsymbol{t}_s}{2} - I\frac{\boldsymbol{t}_s}{2} = 1, \end{aligned} \tag{4.55}$$

where $\widetilde{\boldsymbol{M}}$ is the conjugate motor, and 1 is the identity of the motor multiplication. Now, using Eq. (4.49) and considering the unit motor magnitude, we find two useful properties, expressed by

$$\begin{aligned} |\boldsymbol{M}| = \boldsymbol{M}\widetilde{\boldsymbol{M}} = (\boldsymbol{R}_s + I\boldsymbol{R}'_s)(\widetilde{\boldsymbol{R}}_s + I\widetilde{\boldsymbol{R}}'_s), \\ = \boldsymbol{R}_s\widetilde{\boldsymbol{R}}_s + I(\boldsymbol{R}'_s\widetilde{\boldsymbol{R}}_s + \boldsymbol{R}_s\widetilde{\boldsymbol{R}}'_s) = 1. \end{aligned} \tag{4.56}$$

These equations require the following constraints:

$$\boldsymbol{R}_s\widetilde{\boldsymbol{R}}_s = 1, \tag{4.57}$$

$$\boldsymbol{R}'_s\widetilde{\boldsymbol{R}}_s + \boldsymbol{R}_s\widetilde{\boldsymbol{R}}'_s. \tag{4.58}$$

Now we can show that the combination of two rigid motions can be expressed using two consecutive motors. The resultant motor describes the overall displacement, namely,

$$\begin{aligned} \boldsymbol{M}_c = \boldsymbol{M}_a\boldsymbol{M}_b = (\boldsymbol{R}_{s_a} + I\boldsymbol{R}'_{s_a})(\boldsymbol{R}_{s_b} + I\boldsymbol{R}'_{s_b}), \\ = \boldsymbol{R}_{s_a}\boldsymbol{R}_{s_b} + I(\boldsymbol{R}_{s_a}\boldsymbol{R}'_{s_b} + \boldsymbol{R}'_{s_a}\boldsymbol{R}_{s_b}), \\ = \boldsymbol{R}_{s_c} + I\boldsymbol{R}'_{s_c}. \end{aligned} \tag{4.59}$$

Note that, on the one hand, pure rotations combine multiplicatively, and, on the other hand, the dual parts containing the translation combine additively.

Using Eq. (4.49), let us express a motor in terms of dual spinors:

$$\begin{aligned} \boldsymbol{M} = \boldsymbol{T}_s\boldsymbol{R}_s = \boldsymbol{R}_s + I\boldsymbol{R}'_s, \\ = (a_0 + a_1e_2e_3 + a_2e_3e_2 + a_3e_2e_1), \\ + I(b_0 + b_1e_2e_3 + b_2e_3e_2 + b_3e_2e_1), \\ = (a_0 + \boldsymbol{a}) + I(b_0 + \boldsymbol{b}). \end{aligned} \tag{4.60}$$

We can use another notation to enhance the components of the real and dual parts of the motor, as follows:

$$\boldsymbol{M} = (a_0, \boldsymbol{a}) + I(b_0, \boldsymbol{b}). \tag{4.61}$$

Here, each term within the brackets consists of a scalar part and a 3D bivector.

A motor expressed in terms of a translator and a rotor is manipulated similarly as in the case of a rotor, from the left and its conjugate from the right. These left and right operations, called *motor reflections*, are used to build an automorphism equivalent to the screw. Yet, by conjugating only the rotor or only the translator for the second reflection, we can derive different types of automorphisms.

By changing the sign of the scalar and bivector in the real and dual parts of the motor, we get the following variations:

$$\begin{aligned}
\boldsymbol{M} &= (a_0 + \boldsymbol{a}) + I(b_0 + \boldsymbol{b}) = \boldsymbol{T}_s \boldsymbol{R}_s, \\
\widetilde{\boldsymbol{M}} &= (a_0 - \boldsymbol{a}) + I(b_0 - \boldsymbol{b}) = \widetilde{\boldsymbol{R}}_s \widetilde{\boldsymbol{T}}_s, \\
\bar{\boldsymbol{M}} &= (a_0 + \boldsymbol{a}) - I(b_0 + \boldsymbol{b}) = \boldsymbol{R}_s \widetilde{\boldsymbol{T}}_s, \\
\bar{\widetilde{\boldsymbol{M}}} &= (a_0 - \boldsymbol{a}) - I(b_0 - \boldsymbol{b}) = \widetilde{\boldsymbol{R}}_s \boldsymbol{T}_s.
\end{aligned} \tag{4.62}$$

The first, second, and fourth versions will be used for modeling the motion of points, lines, and planes, respectively.

Using Eq. (4.62), it is now a straightforward matter to compute the expressions for the individual components:

$$\begin{aligned}
a_0 &= \frac{1}{4}(\boldsymbol{M} + \widetilde{\boldsymbol{M}} + \bar{\boldsymbol{M}} + \bar{\widetilde{\boldsymbol{M}}}), \\
I b_0 &= \frac{1}{4}(\boldsymbol{M} + \widetilde{\boldsymbol{M}} - \bar{\boldsymbol{M}} - \bar{\widetilde{\boldsymbol{M}}}), \\
\boldsymbol{a} &= \frac{1}{4}(\boldsymbol{M} - \widetilde{\boldsymbol{M}} + \bar{\boldsymbol{M}} - \bar{\widetilde{\boldsymbol{M}}}), \\
I\boldsymbol{b} &= \frac{1}{4}(\boldsymbol{M} - \widetilde{\boldsymbol{M}} - \bar{\boldsymbol{M}} + \bar{\widetilde{\boldsymbol{M}}}).
\end{aligned} \tag{4.63}$$

4.5.4 The Klein Manifold

Given two homogeneous points in $G_{3,1}$

$$\boldsymbol{X} = x_1 e_1 + y_1 e_2 + z_1 e_3 + e_4 = \mathbf{x}_e + e_4, \tag{4.64}$$

$$\boldsymbol{Y} = x_2 e_1 + y_2 e_2 + z_2 e_3 + e_4 = \mathbf{y} + e_4, \tag{4.65}$$

the line in $G_{4,1}$ crossing these points is given by

$$\begin{aligned}\boldsymbol{L} &= (x_1 - x_2)e_{23} + (y_1 - y_2)e_{31} + (z_1 - z_2)e_{12} + \\ &\quad \cdots + (y_1 z_2 - y_2 z_1)e_{\infty 1} + (x_2 z_1 - x_1 z_2)e_{\infty 2} + (x_1 y_2 - x_2 y_1)e_{\infty 3}, \\ &= I_e(\mathbf{x}_e - \mathbf{y}_e) + e_\infty(\mathbf{x}_e \times \mathbf{y}_e), \\ &= p_{23}e_{23} + p_{31}e_{31} + p_{12}e_{12} + p_{\infty 1}e_{\infty 1} + p_{\infty 2}e_{\infty 2} + p_{\infty 3}e_{\infty 3} = \boldsymbol{n} + \boldsymbol{m},\end{aligned} \tag{4.66}$$

where the coefficients p_{ij} are called the Plücker coordinates, and $\boldsymbol{n}$ and $\boldsymbol{m}$ stand for the line orientation and its momentum, respectively. The later represents a plane which crosses the origin and where the line lies. Due to the homogeneous points $\boldsymbol{X}$ and $\boldsymbol{Y}$, the line is a projective entity as well, which corresponds to a point in a $\mathbb{R}^6$ or a 5D projective space $\mathbb{PR}^5$. Thus, for a nonzero scalar α $\boldsymbol{L} = \alpha\boldsymbol{L}$. Since $\boldsymbol{n} \cdot \boldsymbol{m} = 0$, we obtain the quadratic equation in terms of the Plücker coordinates

$$p_{23}p_{\infty 1} + p_{31}p_{\infty 2} + p_{12}p_{\infty 3} = 0, \tag{4.67}$$

which is a homogeneous equation of degree 2, and hence its solution lies on a 4D quadric hypersurface in $\mathbb{PR}^5$ called Klein quadric. Points of $\mathbb{PR}^5$ not on the quadric cannot be lines. On the other hand, no line in $\mathbb{R}^3$ has $\boldsymbol{n} = 0$; hence, there is a 2D plane in the quadric defined by $\boldsymbol{n} = 0$, whose points are not real points and correspond to lines at infinity. If $\boldsymbol{n} \neq 0$ and $\boldsymbol{m} = 0$ represent lines passing through the origin, summarizing lines in $\mathbb{R}^3$ are points in one-to-one correspondence to point on the 4D Klein quadric excluding the points on the 2D plane.

4.5.5 Reciprocal Screws

The action of a motor (Lie group $SE(3)$) on the lines expressed in terms of the Plücker coordinates is given by

$$\boldsymbol{L}' = \boldsymbol{n}' + \boldsymbol{m}' = \boldsymbol{M}\boldsymbol{L}\widetilde{\boldsymbol{M}}, \tag{4.68}$$

where $\boldsymbol{M} = \boldsymbol{T}\boldsymbol{R} = \boldsymbol{R} + e_\infty \boldsymbol{R}'$. The Klein quadric is invariant under a general rotation and translation or the $SE(3)$ Lie group, due to the isometry

$$\boldsymbol{n}' \cdot \boldsymbol{m}' = (\boldsymbol{R}\boldsymbol{n}\widetilde{\boldsymbol{R}}) \cdot (\boldsymbol{R}\boldsymbol{n}\widetilde{\boldsymbol{R}}' + \boldsymbol{R}'\boldsymbol{n}\widetilde{\boldsymbol{R}} + \boldsymbol{R}'\boldsymbol{m}\widetilde{\boldsymbol{R}}) = \boldsymbol{n} \cdot \boldsymbol{m} = 0. \tag{4.69}$$

Hence, the action of the group $SE(3)$ or the rigid transformation will preserve the Klein quadric.

This shows that $\boldsymbol{n} \cdot \boldsymbol{m}$ is an invariant quadratic form. So given a quadratic form, it can be rewritten as a symmetric bilinear form. For the previous case, the corresponding invariant symmetric bilinear form reads

$$\begin{bmatrix} \mathbf{n}_1^T & \mathbf{m}_1^T \end{bmatrix} \begin{bmatrix} \boldsymbol{O} & I_3 \\ I_3 & \boldsymbol{O} \end{bmatrix} \begin{bmatrix} \boldsymbol{n}_2 \\ \boldsymbol{m}_2 \end{bmatrix} = \boldsymbol{n}_1 \cdot \boldsymbol{m}_2 + \boldsymbol{m}_1 \cdot \boldsymbol{n}_2. \tag{4.70}$$

This form is called the reciprocal product of screws.

A pair of lines meets at point, if their reciprocal product is zero, so we recognize that the screw lines are reciprocal to each other. Parallel lines meet at a point at infinity. All the points at infinity lie on a plane at infinity. There are two different planes lying entirely within the Klein quadric. On α planes lie a pencil of lines which crosses a point in $\mathbb{R}^3$, while β planes are lines on a plane in $\mathbb{R}^2$. In each of the α or β planes lie a pencil of lines crossing at either the origin, a point in $\mathbb{R}^3$, or a point at infinity. Those points are called the projective split of the pencils.

An α plane meets a β plane in a line. In $\mathbb{R}^3$, this line is in the Klein quadric, and it corresponds to the set of line in a plane passing through a point. Such configuration of lines is often named a plane star or a plane pencil of lines. α planes consisting of parallel lines meet at a point at infinity. Parallel lines of two α planes meet a common line at infinity. If an α-plane and a β-plane do meet, then they meet in a line which is on the Klein quadric.

4.5.6 The Study Manifold

The German mathematician Edward Study during late nineteenth and early twentieth centuries interpreted that the group $SE(3)$ of rigid motions can be seen as a quadric hypersurface embedded in a projective space of dimension seven $\mathbb{PR}^7$, that is, as a six-dimensional, degree 2, projective variety. Where every rigid motion can be identified one-to-one with a point in an open set of this quadric, the open set encompasses the entire quadric minus one of its 3-plane generators.

According to Eq. (4.44), any given rigid body motion can be represented by a motor

$$\begin{aligned} \boldsymbol{M} &= \boldsymbol{T}\boldsymbol{R} = \boldsymbol{R} + e_\infty \boldsymbol{R}', \\ &= x_0 + x_1 e_{23} + x_2 e_{31} + x_3 e_{12} + e_\infty (y_0 + y_1 e_{23} + y_2 e_{31} + y_3 e_{12}). \end{aligned} \tag{4.71}$$

That is, the group of these unit motors double covers the group of rigid body motions. This is easy to understand as both correspond to the same rigid motion, since $\boldsymbol{M}$ and $-\boldsymbol{M}$ correspond to the same rigid transformation. Due to this reason, instead of taking the coefficients $(x_0, x_1, x_2, x_3, y_0, y_1, y_2, y_3)$ as coordinates in $\mathbb{R}^8$, we represent the motors instead in the projective space $\mathbb{PR}^7$ with homogeneous coordinates given by ratios $(x_0 : x_1 : x_2 : x_3 : y_0 : y_1 : y_2 : y_3)$. Points represented in terms of homogeneous coordinates are unaffected by multiplication by a nonzero scalar, i.e., is the class $\{M\}$ fulfills $\boldsymbol{M} = \alpha \boldsymbol{M}$. Hence, in this projective space, we have identified the group elements. Every group element $\boldsymbol{M}$ must satisfy

$$\boldsymbol{R} \cdot \boldsymbol{R}' = x_0 y_0 + x_1 y_1 + x_2 y_2 + x_3 y_3 = 0. \tag{4.72}$$

This equation defines a six-dimensional quadric in $\mathbb{RP}^7$ called the Study quadric. In passing from motors to this projective representation, one gets some unwanted points. For any point in the quadric, one can derive a rigid body transformation by dividing the homogeneous coordinates by

$$\pm\sqrt{x_0^2 + x_1^2 + x_2^2 + x_3^2}. \tag{4.73}$$

We should note that points in the quadric not corresponding to rigid transformations are those for which Eq. (4.73) is zero. These exceptional points lie on the 3D plane $x_0 = x_1 = x_2 = x_3 = 0$. Thus, one can claim that the group of rigid motions is isomorphic to the Study quadric with this 3D plane deleted.

4.6 4D Geometric Algebra for Projective 3D Space

To this point, we have dealt with transformations in three-dimensional space. When we use homogeneous coordinates, we increase the dimension of the vector space by one. As a result, the transformation of 3D motion becomes linear. Let us now model the projective 3D space P^3. This space corresponds to the homogeneous extended space R^4. In real applications, it is important to regard the signature of the modeled space to facilitate the computations. In the case of the modeling of the projective plane using homogeneous coordinates, we adopt $G_{3,0,0}$ of the ordinary space, E^3, which has the standard Euclidean signature. For the four-dimensional space R^4, we are forced to adopt the same signature as in the case of the Euclidean space. This geometric algebra $G_{1,3,0}$ is spanned with the following basis:

$$\underbrace{1}_{scalar},\ \underbrace{e_k}_{4\ vectors},\ \underbrace{e_2e_3, e_3e_1, e_1e_2, e_4e_1, e_4e_2, e_4e_3}_{6\ bivectors},\ \underbrace{Ie_k}_{4\ pseudovectors},\ \underbrace{I}_{unit\ pseudoscalar} \tag{4.74}$$

where $e^2 = +1$, $e_k^2 = -1$ for $k = 1, 2, 3$. The unit pseudoscalar is $I = e_1e_2e_3e_4$, with

$$I^2 = (e_1e_2e_3e_4)(e_1e_2e_3e_4) = -(e_3e_4)(e_4) = -1. \tag{4.75}$$

The geometric algebras $G_{3,0,0}$ and $G_{1,3,0}$ will be used for the geometric modeling of the image plane and the visual 3D space. In space–time algebra, the fourth basis vector e_4 of $G_{1,3,0}$ is selected as the time axis for applications of the *projective split* [12]. This helps to associate multivectors of the 4D space with multivectors of the 3D space. The role and use of the projective split for a variety of problems involving the algebra of incidence will also be discussed in Chap. 7.

4.7 Conclusion

This chapter gives an outline of geometric algebra. In particular, it explains the geometric product and the meaning of multivectors. The geometric approach adopted is a version of Clifford algebra, and it is used in the whole book as unifying language for the design of artificial perception–action systems. The chapter presents the geometric algebras of the plane and 3D space, where we can find the planar and 3D quaternions. Finally, we introduce 4D geometric algebras useful for computations involving dual quaternions (motors) or projective geometry. The chapter offers various exercises, so that the reader can start to learn to compute in Clifford algebra in an easy manner.

4.8 Exercises

4.1 Using CLICAL [13] or eClifford [14] in $G_{3,0,0}$, rotate the vector $\boldsymbol{r} = 2e_1 + 3e_2 + 2e_3$ about the axis $\boldsymbol{n} = 1.4e_1 + 1.9e_2$ with angle $\theta = |\boldsymbol{n}|$. Since $|\theta| \leq \phi$, the rotation is well defined.

4.2 Two consecutive rotations in $G_{3,0,0}$, one about the axis $\boldsymbol{a}$ with the angle $\alpha = |\boldsymbol{a}|$ and another about the axis $\boldsymbol{b}$ with the angle $\beta = |\boldsymbol{b}|$, are equivalent to the rotation about an axis $\boldsymbol{c}$. Prove this statement using the Rodriguez formula,

$$\boldsymbol{c}' = \frac{\boldsymbol{a}' + \boldsymbol{b}' - \boldsymbol{a}' \times \boldsymbol{b}'}{1 - \boldsymbol{a}' \cdot \boldsymbol{b}'},$$

where $\boldsymbol{a}' = \frac{\boldsymbol{a}}{\alpha} \tan(\frac{\alpha}{2})$ and $\alpha = |\boldsymbol{a}|$. (*Hint*: Compare the scalar and bivector terms of $e^{\frac{\gamma}{2} I \boldsymbol{c}} = e^{\frac{\alpha}{2} I \boldsymbol{a}} e^{\frac{\beta}{2} I \boldsymbol{b}}$.)

4.3 Using the vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ of $G_{3,0,0}$, prove that the rotation of $\boldsymbol{a}$ to $\boldsymbol{b}$ can be represented by the rotor

$$\boldsymbol{R} = (\boldsymbol{a}\boldsymbol{b})^{\frac{1}{2}} = \frac{(\boldsymbol{a} + \boldsymbol{b})\boldsymbol{b}}{|\boldsymbol{a} + \boldsymbol{b}|} = \frac{\boldsymbol{a}(\boldsymbol{b} + \boldsymbol{a})}{|\boldsymbol{b} + \boldsymbol{a}|}.$$

Since the norm $|\boldsymbol{a} + \boldsymbol{b}| = [2(1 + \boldsymbol{a} \cdot \boldsymbol{b})]^{\frac{1}{2}}$ is not relevant, you can write $\boldsymbol{R}$ as follows:

$$\boldsymbol{R} \doteq (\boldsymbol{a} + \boldsymbol{b})\boldsymbol{b} = \boldsymbol{a}(\boldsymbol{b} + \boldsymbol{a}) = 1 + \boldsymbol{a}\boldsymbol{b}.$$

Also prove this equation. You can interpret the symbol $\doteq$ as a projective identity or an identity up to a scalar factor. (*Hint*: Each rotation can be represented by two reflections.)

4.4 Given the bivector $\boldsymbol{B}$ in the $\boldsymbol{a} \wedge \boldsymbol{b}$ plane and the angle θ between the vectors $\boldsymbol{a}$ and $\boldsymbol{b}$, prove that the rotor

$$\boldsymbol{R} = \frac{1 + \boldsymbol{ba}}{|\boldsymbol{a} + \boldsymbol{b}|}$$

can also be written as $e^{-\frac{\theta}{2}\boldsymbol{B}}$.

4.5 Check by hand, if Eq. 4.46 are correct. Note that the equations have a similar format and that by the equation for the motor the angle is changed to a dual angle and the rotation axis $\boldsymbol{n}$ is changed to a screw line $\boldsymbol{l} = \boldsymbol{n} + I\boldsymbol{m}$.

4.6 Given the rotors $\boldsymbol{R}_1 = e^{-\frac{\alpha}{2}\boldsymbol{B}_1}$ and the rotor $\boldsymbol{R}_2 = e^{-\frac{\alpha}{2}\boldsymbol{B}_2}$, show that their product is

$$\boldsymbol{R}_3 = \boldsymbol{R}_2\boldsymbol{R}_1 = e^{-\frac{\alpha}{2}\boldsymbol{B}_3}.$$

4.7 Show that for an orthonormal bivector basis, the structure constants C^i_{jk} of the Lie algebra of the 3-D rotation group are simply $-\epsilon_{ijk}$.

4.8 Given the bivectors $\boldsymbol{X}$, $\boldsymbol{Y}$, and $\boldsymbol{Z}$, prove the Jacobi identity

$$(\boldsymbol{X} \times \boldsymbol{Y}) \times \boldsymbol{Z} + (\boldsymbol{Z} \times \boldsymbol{X}) \times \boldsymbol{Y} + (\boldsymbol{Y} \times \boldsymbol{Z}) \times \boldsymbol{X} = 0.$$

4.9 Lie algebra: The bivector generators of the unitary group are

$$\begin{aligned} E_{ij} &= e_i e_j + f_i f_j, \quad (i < j = 1, \ldots, n), \\ F_{ij} &= e_i f_j + f_i e_j, \quad (i < j = 1, \ldots, n), \\ J_i &= e_i f_i. \end{aligned}$$

Also show that this algebra is closed under the commutator product and give some examples as well, see [15, Chap. 5, Sect. 5.4].

4.10 Two consecutive motors in $G^+_{3,0,1}$, one about the screw axis $\boldsymbol{L}_a$ with the angle $\alpha = |\boldsymbol{a}|$ and translation $\boldsymbol{t}_a$ and another about the screw axis $\boldsymbol{L}_b$ with the angle $\beta = |\boldsymbol{b}|$ and translation $\boldsymbol{t}_b$, are equivalent to the motor about a screw axis $\boldsymbol{L}_c$. Prove this statement using and extension of the Rodriguez formula given in Exercise 4.2. Draw and interpret your results.

4.11 The dihedral angle ϕ between two planes $\hat{\boldsymbol{X}}$ and $\hat{\boldsymbol{Y}}$ is defined as shown in Fig. 4.7a. Given the unit bivectors $bl\hat{d}X$ and $\hat{\boldsymbol{Y}}$ and $\hat{\boldsymbol{z}}$ the unit vector along the intersection line, prove that

$$\hat{\boldsymbol{X}}\hat{\boldsymbol{Y}} = e^{I\boldsymbol{z}},$$

where $\boldsymbol{z} = \theta\hat{\boldsymbol{z}}$.

4.12 Consider the spherical triangle with corners described by three unit vectors coming from the origin to the surface of a unit sphere, see Fig. 4.7b. The lengths of the three arcs are given by $|\boldsymbol{A}|$, $|\boldsymbol{B}|$, and $|\boldsymbol{C}|$, prove the following formulas:

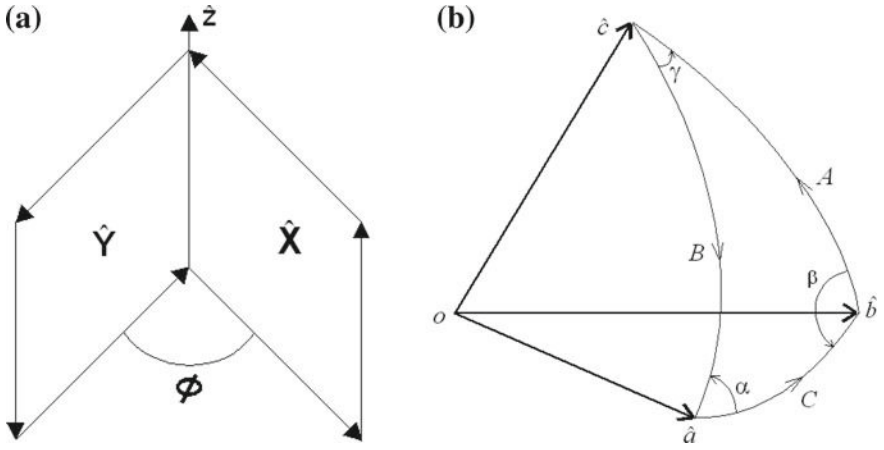


Fig. 4.7 **a** Dihedral angle ϕ between two planes. **b** The spherical triangle

$$\hat{\boldsymbol{a}}\hat{\boldsymbol{b}} = e^{\boldsymbol{C}},$$
$$\hat{\boldsymbol{b}}\hat{\boldsymbol{c}} = e^{\boldsymbol{A}},$$
$$\hat{\boldsymbol{c}}\hat{\boldsymbol{a}} = e^{\boldsymbol{B}}.$$

4.13 Following Exercise 4.12, the angles α, β, and γ between the planes are called the dihedral angles, prove that

$$\hat{\boldsymbol{B}}\hat{\boldsymbol{A}} = e^{I\boldsymbol{c}}, \ \ \|\boldsymbol{c}\| = \gamma,$$
$$\hat{\boldsymbol{C}}\hat{\boldsymbol{B}} = e^{I\boldsymbol{a}}, \ \ \|\boldsymbol{a}\| = \alpha,$$
$$\hat{\boldsymbol{C}}\hat{\boldsymbol{A}} = e^{I\boldsymbol{b}}, \ \ \|\boldsymbol{b}\| = \beta.$$

Also prove that

$$e^{\boldsymbol{C}}e^{\boldsymbol{A}}e^{\boldsymbol{B}} = 1, \ \ e^{I\boldsymbol{c}}e^{I\boldsymbol{b}}e^{I\boldsymbol{a}} = -1.$$

4.14 Following Exercise 4.23, take the scalar part of the equation $e^{-I\boldsymbol{c}} = -e^{I\boldsymbol{b}}e^{I\boldsymbol{a}}$ and prove the cosine law for the angles in spherical trigonometry, namely,

$$\cos(\gamma) = -\cos(\alpha)\cos(\beta) + \sin(\alpha)\sin(\beta)\cos(|\boldsymbol{C}|).$$

Since this formulation is more advantageously as traditional formulations, suggest some applications using this equation.

4.15 Using a sphere, draw the rotated rotor $\boldsymbol{R}_2\boldsymbol{R}_1\widetilde{\boldsymbol{R}}_2$. What it is geometric meaning of this operation with respect to the action of this sequence of rotors acting on a vector?

4.16 Given the bivector $\boldsymbol{B}$ and $\boldsymbol{X}$ and $\boldsymbol{Y}$ two general multivectors, prove that

$$\boldsymbol{B} \times (\boldsymbol{XY}) = (\boldsymbol{B} \times \boldsymbol{X})\boldsymbol{Y} + \boldsymbol{X}(\boldsymbol{B} \times Y).$$

Hence, show that

$$\boldsymbol{B} \times (\boldsymbol{v} \wedge \boldsymbol{V}_r) = (\boldsymbol{B} \cdot \boldsymbol{v}) \wedge \boldsymbol{V} + \boldsymbol{v} \wedge (\boldsymbol{B} \times \boldsymbol{V}_r).$$

By using this result, establish the fact that the operation of commuting with a bivector is grade preserving.

4.17 Given a linear function $f(\boldsymbol{x})$ and an orthonormal frame $\{e_k\}$, one can form the following matrix:

$$f_i j = e_i \cdot f(e_j).$$

Prove that the matrix $\bar{f}(\boldsymbol{x})$ is nothing else as the transport matrix. Furthermore, prove that the product transformations $h = fg$ is determined by the multiplication of f_{ij} and g_{ij}.

4.18 Explain the geometric meaning of the projection $\mathtt{P}_R(\boldsymbol{x}) = (\boldsymbol{x} \cdot \boldsymbol{R})\boldsymbol{R}^{-1}$.

4.19 In $\mathbb{R}^{4,0}$ with an associated orthonormal basis $\{e_i\}_{i=1}^4$, perform a rotation in the plane $e_1 \wedge e_2$, followed with a rotation in the plane $e_3 \wedge e_4$. Compute the rotor of this rotor composition and show that this is the exponent of bivector, not a 2-blade. The resulting rotor is not of a simple form consisting of a scalar plus a 2-blade or even scalar plus a bivector, why?

References

1. Kantor, I. L., & Solodovnikov, A. S. (1989). *Hypercomplex numbers: An elementary introduction to algebras*. New York: Springer-Verlag.
2. Yaglom, M. (1968). *Complex numbers in geometry*. Leicester, UK: Academic Press.
3. Clifford, W. K. (1873). Preliminary sketch of bi-quaternions. *Proceedings of the London Mathematical Society*, *4*, 381–395.
4. Study, E. (1903). *Geometrie der Dynamen*. Leipzig.
5. Rooney, J. (1978). On the three types of complex number and planar transformations. *Environment and Planning B*, *5*, 89–99.
6. Dodwell, P. C. (1983). The Lie transformation group model of visual perception. *Perception and Psychophysics*, *34*(1), 1–16.
7. Hoffman, W. C. (1966). The Lie algebra of visual perception. *Journal of Mathematical Psychology*, *3*, 65–98.
8. Hamilton, W. R. (1853). *Lectures on quaternions*. Dublin: Hodges and Smith.
9. Hamilton, W. R. (1866). *Elements of quaternions*. London: Longmans Green. New York: Chelsea (1969).
10. Bülow, T. (1999). *Hypercomplex Fourier transforms*. Ph.D. Thesis, Computer Science Institute, Christian Albrechts Universität, Kiel.
11. Chernov, V. M. (1995). Discrete orthogonal transforms with data representation in composition algebras. In *Scandinavian Conference on Image Analysis* (pp. 357–364).
12. Lasenby, J., Bayro-Corrochano, E. J., Lasenby, A., & Sommer, G. (1996). A new methodology for computing invariants in computer vision. In *IEEE Proceedings of the International Conference on Pattern Recognition (ICPR'96)* (Vol. I, pp. 393–397), Vienna, Austria.

13. Lounesto, P. (1987). CLICAL software packet and user manual. Helsinki University of Technology of Mathematics, Research report A248.
14. Ablamowicks, R. eCLIFFORD Software packet using Maple for Clifford algebra. Computations. http://math.tntech.edu/rafal.
15. Bayro-Corrochano, E. (2018). *Geometric algebra applications vol. I: Computer vision, graphics and neurocomputing*. Springer-Verlag.

Chapter 5
Kinematics of the 2D and 3D Spaces

5.1 Introduction

This chapter presents the geometric algebra framework for dealing with 3D kinematics. The reader will see the usefulness of this mathematical approach for applications in computer vision and kinematics. We start with an introduction to 4D geometric algebra for 3D kinematics. Then we reformulate, using 3D and 4D geometric algebras, the classic model for the 3D motion of vectors. Finally, we compare both models, that is, the one using 3D Euclidean geometric algebra and our model, which uses 4D motor algebra.

5.2 Representation of Points, Lines, and Planes Using 3D Geometric Algebra

The modeling of points, lines, and planes in 3D Euclidean space will be done using the Euclidean geometric algebra $G_{3,0,0}$, where the pseudoscalar $I^2 = -1$. A point in 3D space represents a position and thus can be simply spanned using the vector basis of $G_{3,0,0}$:

$$\boldsymbol{x} = xe_1 + ye_2 + ze_3, \tag{5.1}$$

where $x,\ y,\ z \in \mathcal{R}$.

In classical vector calculus, a line is described by a position vector $\boldsymbol{x}$ that touches any point of the line and by a vector $\boldsymbol{n}$ for the line direction, that is, $\boldsymbol{l} = \boldsymbol{x} + \alpha\boldsymbol{n}$, where $\alpha \in \mathcal{R}$. In geometric algebra, we employ a multivector concept, and we can thus compactly represent in $G_{3,0,0}$ any line, using a vector $\boldsymbol{n}$ for its direction and a bivector $\boldsymbol{m}$ for the orientation of the plane within which the line lies. Thus,

$$\boldsymbol{l} = \boldsymbol{n} + \boldsymbol{x} \wedge \boldsymbol{n} = \boldsymbol{n} + \boldsymbol{m}. \tag{5.2}$$

© Springer Nature Switzerland AG 2020

E. Bayro-Corrochano, *Geometric Algebra Applications Vol. II*,
https://doi.org/10.1007/978-3-030-34978-3_5

Note that the moment bivector $\boldsymbol{m}$ is computed as the outer product of the position vector $\boldsymbol{x}$ and the line direction vector $\boldsymbol{n}$. We can also compute $\boldsymbol{m}$ as the dual of a vector, that is, $I\boldsymbol{x} \times \boldsymbol{n} = \boldsymbol{m}$.

The representation of the plane is even more striking. The plane is a geometric entity one grade higher than the line, so we would expect that the multivector representation of the plane would be a natural multivector grade extension from that of the line. In classical vector calculus, a plane is described in terms of *Hesse distance*, which represents the distance from the origin to the plane, and a vector which indicates the plane orientation, that is, $\{d,\ \boldsymbol{n}\}$. Note that this description is composed of two separate attributes which comes from the equation: $n_x x + n_y y + n_z z - d = \mathbf{n}^T\mathbf{x} - h = 0$. Once again, using geometric algebra we can express the plane more compactly and with clearer geometric sense. In $G_{3,0,0}$, for example, the extension of the line expression to a plane would be expressed in terms of a bivector $\boldsymbol{n}$ and a trivector Id,

$$\boldsymbol{h} = \boldsymbol{n} + \boldsymbol{x} \wedge \boldsymbol{n} = \boldsymbol{n} + Id, \tag{5.3}$$

where the bivector $\boldsymbol{n}$ indicates the plane orientation, and the outer product of the position vector $\boldsymbol{x}$ and the bivector $\boldsymbol{n}$ builds a trivector which can be expressed using Hesse distance, a scalar value, and the unit pseudoscalar I. Note that the trivector represents a volume, whereas the scalar d represents the Hesse distance. Figure 5.1 compares the different representations of points, lines, and planes using classical vector calculus, Euclidean geometric algebra for $G_{3,0,0}$, and motor algebra for $G^+_{3,0,1}$.

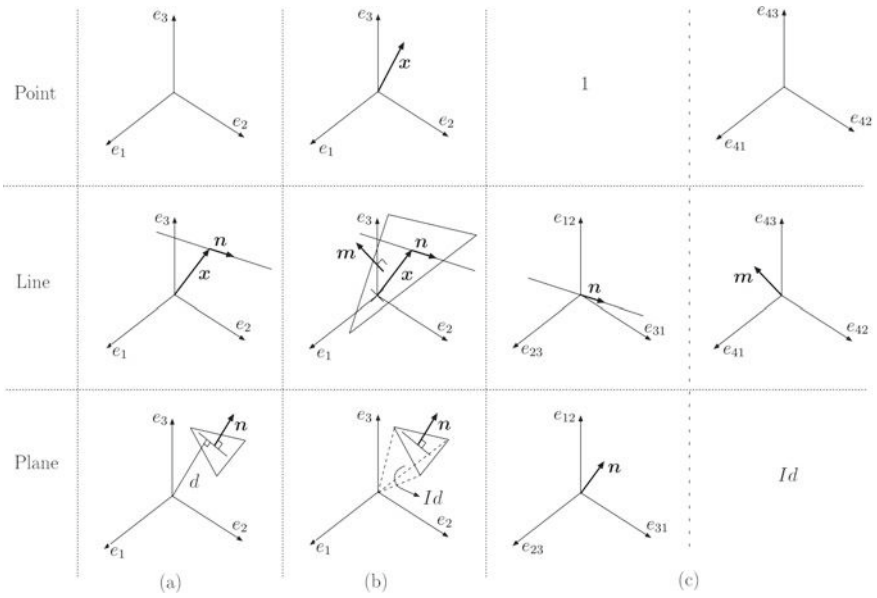


Fig. 5.1 Representations of **a** points, **b** lines, and **c** planes using vector calculus, $G_{3,0,0}$, and $G^+_{3,0,1}$

5.3 Representation of Points, Lines, and Planes Using Motor Algebra

In this section, we will model points, lines, and planes in 4D space using the special algebra of the motors $G^+_{3,0,1}$, which spans in 4D the line space using a bivector basis.

For the case of the point representation, we proceed by embedding a 3D point on the hyperplane $X_4 = 1$, so that the equation of the point $\boldsymbol{X} \in G^+_{3,0,1}$ reads

$$\begin{aligned} \boldsymbol{X} &= 1 + x_1 e_4 e_1 + x_2 e_4 e_2 + x_3 e_4 e_3, \\ &= 1 + I(x_1 e_2 e_3 + x_2 e_3 e_1 + x_3 e_1 e_2), \\ &= 1 + I\boldsymbol{x}, \end{aligned} \tag{5.4}$$

or $\boldsymbol{X} = (1, 0) + I(0, \boldsymbol{x})$. We can see that in this expression the real part consists of the scalar 1 and the dual part of a 3D bivector.

Since we are working in the algebra $G^+_{3,0,1}$ spanned only by bivectors and scalars, we see that this special geometric algebra is the most appropriate system for line modeling. Unlike the line representation, the point and the plane are in some sense asymmetric representations with respect to the scalar and bivector parts. Let us now rewrite the line Eq. (5.2) of $G_{3,0,0}$ in the degenerated geometric algebra $G^+_{3,0,1}$. We can express the vector and the dual vector of Eq. (5.2) in $G^+_{3,0,1}$ as a bivector and a dual bivector. Since the product of the unit pseudoscalar $I = e_1 e_2 e_3 e_4$ and any dual bivectors built from the basis $\{e_4 e_1,\ e_4 e_2,\ e_4 e_3\}$ is zero, we must select the bivector basis $\{e_2 e_3,\ e_3 e_1,\ e_1 e_2\}$ for representing the line

$$\boldsymbol{L} = \boldsymbol{n} + I\boldsymbol{m}. \tag{5.5}$$

In this case, the bivectors for the line direction and the moment are computed using two bivector points, $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, lying on the line, as follows:

$$\begin{aligned} \boldsymbol{n} &= (\boldsymbol{x}_2 - \boldsymbol{x}_1) = (x_{21} - x_{11}) e_2 e_3 + (x_{22} - x_{12}) e_3 e_1 + (x_{23} - x_{13}) e_1 e_2, \\ &= L_{n_1} e_2 e_3 + L_{n_2} e_3 e_1 + L_{n_3} e_1 e_2, \\ \boldsymbol{m} &= \boldsymbol{x}_1 \times \boldsymbol{x}_2, \\ &= (x_{12} x_{23} - x_{13} x_{22}) e_2 e_3 + (x_{13} x_{21} - x_{11} x_{23}) e_3 e_1 + \cdots + \\ &\qquad + (x_{11} x_{22} - x_{12} x_{21}) e_1 e_2, \\ &= L_{m_1} e_2 e_3 + L_{m_2} e_3 e_1 + L_{m_3} e_1 e_2. \end{aligned} \tag{5.6}$$

This line representation using dual numbers is easy to understand and to manipulate algebraically, and it is fully equivalent to the representation in terms of *Plücker coordinates*. Using bracket notation, the line equation becomes $\boldsymbol{L} \equiv (0, \boldsymbol{n}) + I(0, \boldsymbol{m})$, where $\boldsymbol{n}$ and $\boldsymbol{m}$ are spanned with a 3D bivector basis.

For the equation of the plane, we proceed in a similar manner as for Eq. (5.3). We represent the orientation of the plane via the bivector $\boldsymbol{n}$ and the outer product

between a bivector touching the plane and its orientation $\boldsymbol{n}$. This outer product results in a quatrivector, which we can express as the Hesse distance $d = (\boldsymbol{x} \cdot \boldsymbol{n})$ multiplied by the unit pseudoscalar,

$$\boldsymbol{H} = \boldsymbol{n} + \boldsymbol{x} \wedge \boldsymbol{n} = \boldsymbol{n} + I(\boldsymbol{x} \cdot \boldsymbol{n}) = \boldsymbol{n} + Id, \tag{5.7}$$

or $\boldsymbol{H} = (0, \boldsymbol{n}) + I(d, 0)$. Note that the plane equation is the dual of the point equation

$$\boldsymbol{H} = (d + I\boldsymbol{n})^* = (I\boldsymbol{n})^* + (d)^* = \boldsymbol{n} + Id, \tag{5.8}$$

where the plane orientation is given by the unit bivector $\boldsymbol{n}$ and the Hesse distance d by the scalar 1.

5.4 Representation of Points, Lines, and Planes Using 4D Geometric Algebra

We can also represent point, lines, and planes using the entire 4D geometric algebra $G_{3,0,1}$. As opposite to the previous representations which use only bivectors, this representation will also use vectors and the trivector basis. The point expressed in terms of trivectors is given by

$$\boldsymbol{X} = e_1e_2e_3 + x_1e_2e_3e_4 + x_2e_3e_1e_4 + x_3e_1e_2e_4. \tag{5.9}$$

The equation of the line using bivectors basis is exactly the same as that of Eq. (5.5):

$$\begin{aligned}\boldsymbol{L} &= L_{n_1}e_2e_3 + L_{n_1}e_3e_1 + L_{n_3}e_1e_2 + L_{m_1}e_4e_1 + L_{m_2}e_4e_2 + L_{m_3}e_4e_3,\\ &= L_{n_1}e_2e_3 + L_{n_1}e_3e_1 + L_{n_3}e_1e_2 + I(L_{m_1}e_2e_3 + L_{m_2}e_3e_1 + L_{m_3}e_1e_2),\\ &= \boldsymbol{n} + I\boldsymbol{m}.\end{aligned} \tag{5.10}$$

The equation of the plane is spanned using basis vectors in terms of the normal of the plane and the Hessian distance:

$$\boldsymbol{H} = \boldsymbol{n}_xe_1 + \boldsymbol{n}_ye_2 + \boldsymbol{n}_ze_3 + de_4. \tag{5.11}$$

Note that in this equation the multivector bases of the point and plane have been swapped. This equation corresponds to the dual of point Eq. (5.9) and makes use of a vector as the dual of each trivector:

$$\begin{aligned}\boldsymbol{H} &= (de_1e_2e_3 + n_xe_2e_3e_4 + n_ye_3e_1e_4 + n_ze_1e_2e_4)^*,\\ &= d(e_1e_2e_3)^* + n_x(e_2e_3e_4)^* + n_y(e_3e_1e_4)^* + n_z(e_1e_2e_4)^*,\\ &= n_xe_1 + n_ye_2 + n_ze_3 + de_4.\end{aligned} \tag{5.12}$$

Note that the dual operation is actually not carried out via the pseudoscalar $I = e_1e_2e_3e_4$ because this will lead to the square of e_4, which equals zero. In order to explain the relation between Eqs. (5.9) and (5.11), we are simply relating in the dual sense each basis vector with a basis trivector. Since Eqs. (5.4) and (5.7) are the duals of Eqs. (5.9) and (5.11), we can reconsider Fig. 5.1c, now using a trivector coordinate basis for depicting the point Eq. (5.9), and similarly, Fig. 5.1a, now using a vector coordinate basis for the plane Eq. (5.11).

The following sections are concerned with the modeling of the motion of basic geometric entities in 3D and 4D spaces. By comparing these motion models, we will show the power of geometric algebra in the representation and linearizing of the translation transformation achieved in 4D geometric algebra.

5.5 Motion of Points, Lines, and Planes in 3D Geometric Algebra

The 3D motion of a point $\boldsymbol{x}$ in $G_{3,0,0}$ is given by the following equation:

$$\boldsymbol{x}' = \boldsymbol{R}\boldsymbol{x}\tilde{\boldsymbol{R}} + \boldsymbol{t}. \tag{5.13}$$

Using Eq. (5.2), the motion equation of the line can be expressed as follows:

$$\begin{aligned}
\boldsymbol{l}' &= \boldsymbol{n}' + \boldsymbol{m}' = \boldsymbol{n}' + \boldsymbol{x}' \wedge \boldsymbol{n}', \\
&= \boldsymbol{R}\boldsymbol{n}\tilde{\boldsymbol{R}} + (\boldsymbol{R}\boldsymbol{x}\tilde{\boldsymbol{R}} + \boldsymbol{t}) \wedge (\boldsymbol{R}\boldsymbol{n}\tilde{\boldsymbol{R}}), \\
&= \boldsymbol{R}\boldsymbol{n}\tilde{\boldsymbol{R}} + \boldsymbol{R}\boldsymbol{x}\tilde{\boldsymbol{R}} \wedge \boldsymbol{R}\boldsymbol{n}\tilde{\boldsymbol{R}} + \boldsymbol{t} \wedge \boldsymbol{R}\boldsymbol{n}\tilde{\boldsymbol{R}}, \\
&= \boldsymbol{R}\boldsymbol{n}\tilde{\boldsymbol{R}} + \boldsymbol{R}\boldsymbol{x}\tilde{\boldsymbol{R}} \wedge \boldsymbol{R}\boldsymbol{n}\tilde{\boldsymbol{R}} + \frac{\boldsymbol{t}}{2}\boldsymbol{R}\boldsymbol{n}\tilde{\boldsymbol{R}} - \boldsymbol{R}\boldsymbol{n}\tilde{\boldsymbol{R}}\frac{\boldsymbol{t}}{2}, \\
&= \boldsymbol{R}\boldsymbol{n}\tilde{\boldsymbol{R}} + \boldsymbol{R}\boldsymbol{n}\frac{\boldsymbol{t}}{2}\tilde{\boldsymbol{R}} + \frac{\boldsymbol{t}}{2}\boldsymbol{R}\boldsymbol{n}\tilde{\boldsymbol{R}} + \boldsymbol{R}\boldsymbol{m}\tilde{\boldsymbol{R}},
\end{aligned} \tag{5.14}$$

where $\boldsymbol{x}'$ stands for the rotated and shifted position vector, $\boldsymbol{n}'$ stands for the rotated orientation vector, and $\boldsymbol{m}'$ for the new line moment.

The model of the motion of the plane in $G_{3,0,0}$ can be expressed in terms of the multivector Hesse equation (5.3), as follows:

$$\begin{aligned}
\boldsymbol{h}' &= \boldsymbol{n}' + I\boldsymbol{d}' = \boldsymbol{n}' + \boldsymbol{x}' \wedge \boldsymbol{n}', \\
&= \boldsymbol{R}\boldsymbol{n}\tilde{\boldsymbol{R}} + (\boldsymbol{R}\boldsymbol{x}\tilde{\boldsymbol{R}} + \boldsymbol{t}) \wedge (\boldsymbol{R}\boldsymbol{n}\tilde{\boldsymbol{R}}), \\
&= \boldsymbol{R}\boldsymbol{n}\tilde{\boldsymbol{R}} + \boldsymbol{R}\boldsymbol{x}\tilde{\boldsymbol{R}} \wedge \boldsymbol{R}\boldsymbol{n}\tilde{\boldsymbol{R}} + \boldsymbol{t} \wedge \boldsymbol{R}\boldsymbol{n}\tilde{\boldsymbol{R}}, \\
&= \boldsymbol{R}\boldsymbol{n}\tilde{\boldsymbol{R}} + \boldsymbol{t} \wedge \boldsymbol{R}\boldsymbol{n}\tilde{\boldsymbol{R}} + \boldsymbol{R}\boldsymbol{x} \wedge \boldsymbol{n}\tilde{\boldsymbol{R}}, \\
&= \boldsymbol{R}\boldsymbol{n}\tilde{\boldsymbol{R}} + \boldsymbol{t} \wedge \boldsymbol{R}\boldsymbol{n}\tilde{\boldsymbol{R}} + \boldsymbol{R}(I\boldsymbol{d})\tilde{\boldsymbol{R}}, \\
&= \boldsymbol{R}\boldsymbol{n}\tilde{\boldsymbol{R}} + \boldsymbol{t}^* \cdot \boldsymbol{R}\boldsymbol{n}\tilde{\boldsymbol{R}} + I\boldsymbol{d}, \\
&= \boldsymbol{R}\boldsymbol{n}\tilde{\boldsymbol{R}} + I(\boldsymbol{t} \cdot \boldsymbol{R}\boldsymbol{n}\tilde{\boldsymbol{R}} + d),
\end{aligned} \tag{5.15}$$

where $\boldsymbol{n}'$ stands for the rotated bivector plane orientation, $\boldsymbol{x}'$ stands for the rotated and shifted position vector, and d' for the new Hesse distance. Here, we use the concept of duality to claim that $\boldsymbol{t} \wedge \boldsymbol{Rn}\tilde{\boldsymbol{R}} = \boldsymbol{t}^* \cdot \boldsymbol{Rn}\tilde{\boldsymbol{R}} = (I\boldsymbol{t}) \cdot \boldsymbol{Rn}\tilde{\boldsymbol{R}}$.

5.6 Motion of Points, Lines, and Planes Using Motor Algebra

The modeling of the 3D motion of geometric primitives using the motor algebra $G^+_{3,0,1}$ takes place in a 4D space where rotation and translation are multiplicative operators that are applied as multiplicative operators, and the result is that the 3D general motion becomes linear. Having created a linear model, we can then compute simultaneously the unknown rotation and translation. This will be useful for the case of the hand–eye problem, or when we apply the motor extended Kalman filter.

For the modeling of the point motion, we use the point representation of Eq. (5.4) and the motor relations given in Eq. (4.62), with $I^2 = 0$:

$$\begin{aligned}
\boldsymbol{X}' &= 1 + I\boldsymbol{x}' = \boldsymbol{M}\boldsymbol{X}\tilde{\boldsymbol{M}} = \boldsymbol{M}\,(1 + I\boldsymbol{x})\,\tilde{\boldsymbol{M}}, \\
&= \boldsymbol{T}_s\boldsymbol{R}_s\,(1 + I\boldsymbol{x})\,\tilde{\boldsymbol{R}}_s\boldsymbol{T}_s, \\
&= \left(1 + I\frac{\boldsymbol{t}_s}{2}\right)\boldsymbol{R}_s\,(1 + I\boldsymbol{x})\,\tilde{\boldsymbol{R}}_s\left(1 + I\frac{\boldsymbol{t}_s}{2}\right), \\
&= \left(1 + I\frac{\boldsymbol{t}_s}{2}\right)\left(1 + I\boldsymbol{R}_s\boldsymbol{x}\tilde{\boldsymbol{R}}_s\right)\left(1 + I\frac{\boldsymbol{t}_s}{2}\right), \\
&= 1 + I\frac{\boldsymbol{t}_s}{2} + I\boldsymbol{R}_s\boldsymbol{x}\tilde{\boldsymbol{R}}_s + I\frac{\boldsymbol{t}_s}{2}, \\
&= 1 + I\left(\boldsymbol{R}_s\boldsymbol{x}\tilde{\boldsymbol{R}}_s + \boldsymbol{t}_s\right).
\end{aligned} \tag{5.16}$$

Note that the dual part of this equation in 4D space is fully equivalent to Eq. (5.13), which is in 3D space.

The motion of a 3D line or screw motion can be seen as the rotation of the line about the axis line $\boldsymbol{L}_s$ and its translation along this axis line, as depicted in Fig. 5.2. Note that in the figure the line $\boldsymbol{L}_s$ is shifted a distance $\boldsymbol{t}_c$ from the origin. Now, using line Eq. (5.5), we can express the motion of a 3D line as follows:

$$\boldsymbol{L}' = \boldsymbol{n}' + I\boldsymbol{m}' = \boldsymbol{M}(\boldsymbol{L})\tilde{\boldsymbol{M}} = \boldsymbol{T}_s\boldsymbol{R}_s(\boldsymbol{n} + I\boldsymbol{m})\tilde{\boldsymbol{R}}_s\tilde{\boldsymbol{T}}_s. \tag{5.17}$$

This equation can be further expressed purely in terms of rotors, as follows:

$$\boldsymbol{L}' = \left(1 + I\frac{\boldsymbol{t}_s}{\boldsymbol{2}}\right)\boldsymbol{R}_s\,(\boldsymbol{n} + I\boldsymbol{m})\,\tilde{\boldsymbol{R}}_s\left(1 - I\frac{\boldsymbol{t}_s}{2}\right),$$

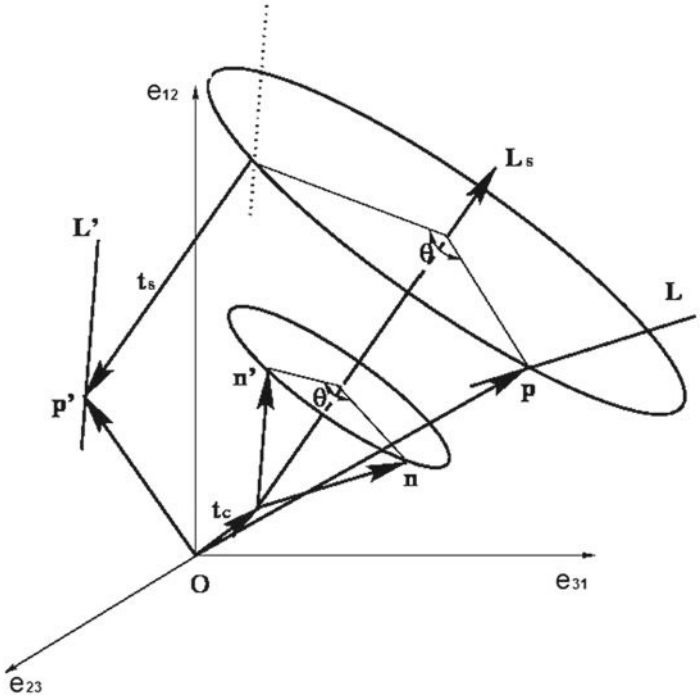


Fig. 5.2 The screw motion of a line

$$
\begin{aligned}
&= \left(1 + I\frac{\boldsymbol{t}_s}{\mathbf{2}}\right)\left(\boldsymbol{R}_s\boldsymbol{n}\tilde{\boldsymbol{R}}_s + I\boldsymbol{R}_s\boldsymbol{m}\tilde{\boldsymbol{R}}_s - I\boldsymbol{R}_s\boldsymbol{n}\tilde{\boldsymbol{R}}_s\frac{\boldsymbol{t}_s}{2}\right), \\
&= \boldsymbol{R}_s\boldsymbol{n}\tilde{\boldsymbol{R}}_s + I\left(-\boldsymbol{R}_s\boldsymbol{n}\tilde{\boldsymbol{R}}_s\frac{\boldsymbol{t}_s}{2} + \frac{\boldsymbol{t}_s}{2}\boldsymbol{R}_s\boldsymbol{n}\tilde{\boldsymbol{R}}_s + \boldsymbol{R}_s\boldsymbol{m}\tilde{\boldsymbol{R}}_s\right), \\
&= \boldsymbol{R}_s\boldsymbol{n}\tilde{\boldsymbol{R}}_s + I\left(\boldsymbol{R}_s\boldsymbol{n}\tilde{\boldsymbol{R}}_s{}' + \boldsymbol{R}_s'\boldsymbol{n}\tilde{\boldsymbol{R}}_s + \boldsymbol{R}_s\boldsymbol{m}\tilde{\boldsymbol{R}}_s\right). \qquad (5.18)
\end{aligned}
$$

Note that in this equation, before we merge the bivector $\frac{\boldsymbol{t}_s}{2}$ with the rotor $\boldsymbol{R}_s$ or $\tilde{\boldsymbol{R}}_s$, the real and the dual parts are fully equivalent with the elements of line Eq. (5.14) of $G_{3,0,0}$.

Equation (5.18) is very useful as a linear algorithm to estimate simultaneous rotation and translation, as in the case of hand–eye calibration [1] or for the algorithm for the motor extended Kalman filter.

The transformation of a plane under a rigid motion in $G^{+}_{3,0,1}$ can be seen as the motion of the dual of the point. Thus, Eq. (5.7) can be utilized to express the motion equation of the plane:

$$
\begin{aligned}
\boldsymbol{H}' &= \boldsymbol{n}' + I d' = \boldsymbol{M}\boldsymbol{H}\tilde{\boldsymbol{M}} = \boldsymbol{M}\left(\boldsymbol{n} + I d\right)\tilde{\boldsymbol{M}}, \\
&= \boldsymbol{T}_s \boldsymbol{R}_s \left(\boldsymbol{n} + I d\right) \tilde{\boldsymbol{R}}_s \boldsymbol{T}_s, \\
&= \left(1 + I\frac{\boldsymbol{t}_s}{\mathbf{2}}\right)\left(\boldsymbol{R}_s \boldsymbol{n} \tilde{\boldsymbol{R}}_s + I d\right)\left(1 + I\frac{\boldsymbol{t}_s}{\mathbf{2}}\right), \\
&= \boldsymbol{R}_s \boldsymbol{n} \tilde{\boldsymbol{R}}_s + I\left(\boldsymbol{R}_s \boldsymbol{n} \tilde{\boldsymbol{R}}_s \frac{\boldsymbol{t}_s}{2} + \frac{\boldsymbol{t}_s}{2}\boldsymbol{R}_s \boldsymbol{n} \tilde{\boldsymbol{R}}_s + d\right), \\
&= \boldsymbol{R}_s \boldsymbol{n} \tilde{\boldsymbol{R}}_s + I\left(\boldsymbol{t}_s \cdot \left(\boldsymbol{R}_s \boldsymbol{n} \tilde{\boldsymbol{R}}_s\right) + d\right).
\end{aligned} \tag{5.19}
$$

Note that the real part and the dual part of this expression are fully equivalent to the bivector and trivector parts of Eq. (5.15) in $G_{3,0,0}$.

5.7 Motion of Points, Lines, and Planes Using 4D Geometric Algebra

For the modeling of the motion of points, lines, and planes in $\mathcal{G}_{3,0,1}$, only the automorphism equivalent of the screw and its conjugate are required.

The motion of the point is given by

$$
\begin{aligned}
\boldsymbol{X}' &= \boldsymbol{M}\boldsymbol{X}\tilde{\boldsymbol{M}} = \boldsymbol{T}_s \boldsymbol{R}_s \left(\boldsymbol{X}\right) \tilde{\boldsymbol{R}}_s \tilde{\boldsymbol{T}}_s, \\
&= \left(1 + I\frac{\boldsymbol{t}_s}{2}\right) \boldsymbol{R}_s \left(e_1 e_2 e_3 + I\boldsymbol{x}\right) \tilde{\boldsymbol{R}}_s \left(1 - I\frac{\boldsymbol{t}_s}{2}\right), \\
&= \left(1 + e_4\frac{\boldsymbol{t}}{2}\right)\left(e_1 e_2 e_3 + I\boldsymbol{R}_s \boldsymbol{x} \tilde{\boldsymbol{R}}_s\right)\left(1 - e_4\frac{\boldsymbol{t}}{2}\right), \\
&= e_1 e_2 e_3 + I\frac{\boldsymbol{t}}{2} + I\boldsymbol{R}_s \boldsymbol{x} \tilde{\boldsymbol{R}}_s + I\frac{\boldsymbol{t}_s}{2}, \\
&= e_1 e_2 e_3 + I\left(\boldsymbol{R}_s \boldsymbol{x} \tilde{\boldsymbol{R}}_s + \boldsymbol{t}_s\right),
\end{aligned} \tag{5.20}
$$

where $\boldsymbol{t}_s = t_x e_2 e_3 + t_y e_3 e_1 + t_z e_1 e_2, \boldsymbol{x} = x_x e_1 + x_y e_2 + x_z e_3$, and $\boldsymbol{t} = t_x e_1 + t_y e_2 + t_z e_3$. Note that the dual part of this equation in 4D space is fully equivalent to Eq. (5.13) in 3D space.

The equation of the motion of the line is exactly the same as Eq. (5.18).

The motion of the plane is given by

$$\begin{aligned}
\boldsymbol{H}' &= \boldsymbol{n}' + e_4 d' = n'_x e_1 + n'_y e_2 + n'_z e_3 + d' e_4 = \boldsymbol{M}\boldsymbol{H}\tilde{\boldsymbol{M}} = \boldsymbol{M}\left(\boldsymbol{n} + e_4 d\right)\tilde{\boldsymbol{M}}, \\
&= \boldsymbol{T}_s \boldsymbol{R}_s \left(\boldsymbol{n} + e_4\right)\tilde{\boldsymbol{R}}_s \tilde{\boldsymbol{T}}_s = \left(1 + I\frac{\boldsymbol{t}_s}{\mathbf{2}}\right)\left(\boldsymbol{R}_s \boldsymbol{n}\tilde{\boldsymbol{R}}_s + e_4 d\right)\left(1 - I\frac{\boldsymbol{t}_s}{\mathbf{2}}\right), \\
&= \boldsymbol{R}_s \boldsymbol{n}\tilde{\boldsymbol{R}}_s + I\frac{\boldsymbol{t}_s}{\mathbf{2}} R_s \boldsymbol{n}\tilde{\boldsymbol{R}}_s - \boldsymbol{R}_s \boldsymbol{n}\tilde{\boldsymbol{R}}_s I\frac{\boldsymbol{t}_s}{\mathbf{2}} + e_4 d, \\
&= \boldsymbol{R}_s \boldsymbol{n}\tilde{\boldsymbol{R}}_s + I\left(\frac{\boldsymbol{t}_s R_s \boldsymbol{n}\tilde{\boldsymbol{R}}_s - \boldsymbol{R}_s \boldsymbol{n}\tilde{\boldsymbol{R}}_s \boldsymbol{t}_s}{\mathbf{2}}\right) + e_4 d, \\
&= \boldsymbol{R}_s \boldsymbol{n}\tilde{\boldsymbol{R}}_s + I\left(\boldsymbol{t}_s \wedge R_s \boldsymbol{n}\tilde{\boldsymbol{R}}_s\right) + e_4 d, \\
&= \boldsymbol{n}' + e_4\left(I_3 \boldsymbol{t}_s\right) \wedge \boldsymbol{n}' + e_4 d = \boldsymbol{n}' + e_4\left(\boldsymbol{t}_s^* \cdot \boldsymbol{n}'\right) + e_4 d, \\
&= \boldsymbol{n}' + e_4\left(\boldsymbol{t} \cdot \boldsymbol{n}' + d\right).
\end{aligned} \tag{5.21}$$

The real and dual parts of this expression are equivalent in a higher dimension to the bivector and trivector parts of Eq. (5.15) in $G_{3,0,0}$.

5.8 Spatial Velocity of Points, Lines, and Planes

This section begins with the classic formulation of the spatial velocity of a rigid body using matrices; in a later subsection, we will represent this velocity using instead the more advantageous techniques of motor algebra.

5.8.1 Rigid Body Spatial Velocity Using Matrices

The 3D motion of a rigid body involves a rotation and a translation and can be computed as a relative motion between world coordinates, fixed frame and a frame attached to the object. In the case of pure rotations, the body frame is set at the origin of the fixed spatial frame. Translations will displace the object frame apart from the spatial frame. The velocity of a single particle of the object is given by

$$v_x(t) = \frac{dx(t)}{dt}. \tag{5.22}$$

Let us first consider the trajectory of a continuous rotational motion given by $R(t) : \mathbb{R} \rightarrow SO(3)$ that satisfies the constraint

$$R(t)R^T(t) = I. \tag{5.23}$$

By computing the derivative of this equation with respect to time t and passing a term to the right, we get a skew-symmetric matrix $\hat{w}(t)$:

$$\dot{R}(t)R^T(t) + R(t)\dot{R}^T(t) = 0 \Rightarrow \dot{R}(t)R^T(t) = -R(t)\dot{R}^T(t) = \hat{w}(t). \quad (5.24)$$

We get the expression for the derivative of the 3D rotation by multiplying both sides of the last equation by $R(t)$:

$$\dot{R}(t) = \hat{w}(t)R(t). \quad (5.25)$$

Considering that at t_0, there is not yet a rotation, $R(t_0) = I$, the first-order approximation to a rotation matrix is given by

$$\dot{R}(t_0 + dt) \approx I + \hat{w}(t_0)dt. \quad (5.26)$$

The linear space of all skew-symmetric matrices is commonly denoted by

$$so(3) \doteq \{\hat{w} \in \mathbb{R}^{3\times 3} | w \in \mathbb{R}^3\}. \quad (5.27)$$

In Eq. (5.25), $R(t)$ can be interpreted as the $state\,transition\,matrix$ of the following linear Ordinary Differential Equation (ODE):

$$\dot{x}(t) = \hat{w}x(t), \quad x(t) \in \mathbb{R}^3, \quad (5.28)$$

whose solution is given by

$$x(t) = e^{\hat{w}}x(0), \quad (5.29)$$

and the matrix $e^{\hat{w}t}$ can be expanded as a McClaurin expansion as follows:

$$e^{\hat{w}t} = I + \hat{w}t + \frac{(\hat{w}t)^2}{2!} + \cdots + \frac{(\hat{w}t)^n}{n!} + \cdots. \quad (5.30)$$

According to Rodriguez' formula related to a rotation matrix, the last equation can also be written in a more compact form as

$$e^{\hat{w}t} = I + \frac{\hat{w}}{||w||}sin(||w||) + \frac{\hat{w}^2}{||w||^2}(1 - \cos(||w||)). \quad (5.31)$$

Due to the uniqueness of the solution of Eq. (5.28) and assuming $R(0) = I$ is the initial condition of Eq. (5.25), we get

$$R(t) = e^{\hat{w}t}R(0) = e^{\hat{w}t}. \quad (5.32)$$

Interestingly enough, we have derived the exponential relation between the linear spaces of $so(3)$ and $SO(3)$

$$\exp : so(3) \rightarrow SO(3) : \quad \hat{w} \mapsto e^{\hat{w}}, \tag{5.33}$$

where the inverse of the exponential map is given by $\hat{w} = \log(R)$.

The motion of a rigid body can be represented as a transformation matrix using homogeneous coordinates. The set of all motion matrices forms a group that is no longer Euclidean; thus, the formulation of the rigid body velocity cannot be done as a straightforward extension of Eq. (5.22) of the velocity of a single particle. One can derive the velocity equations of the rigid body based on the parametrization of motion using homogeneous matrices. Let us represent the trajectory of a body as a time-dependent curve using a homogeneous matrix:

$$g(t) = \begin{bmatrix} R(t) & t(t) \\ 0^T & 1 \end{bmatrix} \in I\!R^{4\times 4}. \tag{5.34}$$

The inverse of $g(t)$ is given by

$$g^{-1}(t) = \begin{bmatrix} R(t) & t(t) \\ 0^T & 1 \end{bmatrix}^{-1} = \begin{bmatrix} R^T & -R^T t(t) \\ 0^T & 1 \end{bmatrix}. \tag{5.35}$$

In analogy to the case of pure rotation (see Eqs. (5.25)–(5.28)), we will obtain a matrix consisting of the instantaneous *spatial angular velocity* component expressed as an antisymmetric matrix $\hat{w}$ and the linear-velocity component v; for this, we start deriving the following identity with respect to time t

$$\begin{aligned} \frac{dg(t)g^{-1}(t)}{dt} &= \frac{dI}{dt} = 0, \\ \dot{g}(t)g^{-1}(t) + g(t)\dot{g}^{-1}(t) &= 0. \end{aligned} \tag{5.36}$$

Now, passing one term to the right, we get a skew-symmetric matrix $\widehat{V}(t)$

$$\dot{g}(t)g^{-1}(t) = -g(t)\dot{g}^{-1}(t) = \hat{V}(t) \in I\!R^{4\times 4}, \tag{5.37}$$

which equals

$$\begin{aligned} \widehat{V} = \dot{g(t)}g^{-1}(t) &= \begin{bmatrix} \dot{R}(t) & \dot{t}(t) \\ 0^T & 0 \end{bmatrix} \begin{bmatrix} R^T(t) & -R^T(t)t(t) \\ 0^T & 1 \end{bmatrix}, \\ &= \begin{bmatrix} \dot{R}(t)R^T(t) & -\dot{R}(t)R^T(t)t(t) + \dot{t}(t) \\ 0^T & 0 \end{bmatrix} = \begin{bmatrix} \hat{w}(t) & v(t) \\ 0^T & 0 \end{bmatrix}. \end{aligned} \tag{5.38}$$

One gets the expression for the derivative of the 3D motion by multiplying both sides of the last equation by $g(t)$:

$$\dot{g}(t) = (\dot{g}(t)g^{-1}(t))g(t) = \widehat{V}g(t). \tag{5.39}$$

$\widehat{V}$ can be viewed as the tangent vector along the curve of $g(t)$ and can be used to approximate $g(t)$ locally:

$$\dot{g}(t+dt) \approx g(t) + \widehat{V}g(t)dt = (I + \widehat{V}dt)g(t). \tag{5.40}$$

The 4×4 matrix of the form of $\hat{V}$ is called a $twist$. The set of all twists builds a linear space denoted by

$$se(3) \doteq \left\{ \hat{V} \in \begin{bmatrix} \hat{w} & v \\ 0^T & 0 \end{bmatrix} \middle| w \in so(3),\ v \in I\!R^3 \right\} \in I\!R^{4\times 4}. \tag{5.41}$$

$se(3)$ is called the tangent space or Lie algebra of the matrix Lie group $SE(3)$.

If, in Eq. (5.39), $\widehat{V}$ is considered constant, we have a time-invariant linear ordinary equation, which can be integrated to get the following expression:

$$g(t) = e^{\widehat{V}t}g(0). \tag{5.42}$$

If we assume the initial condition $g(0) = I$, we obtain

$$g(t) = e^{\widehat{V}t}. \tag{5.43}$$

As a result, we can claim that the exponential map defines a transformation from $se(3)$ to $SE(3)$, namely,

$$exp : se(3) \to SE(3); \quad \widehat{V} \to e^{\widehat{V}}. \tag{5.44}$$

This exponential twist can be further expanded using the McClaurin series

$$e^{\widehat{V}t} = I + \widehat{V}t + \frac{(\widehat{V}t)^2}{2!} + \cdots + \frac{\widehat{V}^n}{n!} + \cdots. \tag{5.45}$$

Using the Rodriguez formula (5.31) and certain additional properties of the exponential matrix, one can establish the following relationship:

$$e^{\widehat{V}} = \begin{bmatrix} e^{\hat{w}} & \frac{(I-e^{\hat{w}})\hat{w}v + ww^Tv}{||w||} \\ 0^T & 1 \end{bmatrix}, \quad \texttt{if} \quad w \neq 0. \tag{5.46}$$

If $w = 0$, the exponential of $\widehat{V}$ becomes

$$e^{\widehat{V}} = \begin{bmatrix} I & v \\ 0^T & 1 \end{bmatrix}. \tag{5.47}$$

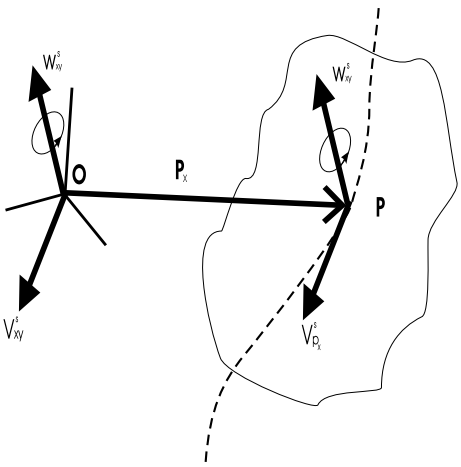


Fig. 5.3 Linear and angular velocities of a body with respect to spatial frame X and body frame Y

Let us now study the rigid motion with respect to a spatial framework X and other Y attached to a rigid body (see Fig. 5.3). Let us again consider the equation of the twist derived above:

$$\widehat{V}^s_{xy} = \begin{bmatrix} \dot{R}_{xy}R^T_{xy} & -\dot{R}_{xy}R^T_{xy}t_{xy} + \dot{t}_{xy} \\ 0^T & 0 \end{bmatrix} = \begin{bmatrix} \hat{w}^s_{xy} & v^s_{xy} \\ 0^T & 0 \end{bmatrix}. \tag{5.48}$$

Note that the physical meaning of the linear-velocity component is not very intuitive, due to the antisymmetric matrix $\dot{R}_{xy}R^T_{xy} = \hat{w}^s_{xy}$; there is a component orthogonal to the actual translation and parallel to the velocity with respect to the object frame, namely, $(\hat{w}^s_{xy}t_{xy} = w^s_{xy} \times t_{xy}) \,||\, \dot{t}_{xy}$ (see Fig. 5.3). We will see below that only with the bivector notation of the motor algebra, we can achieve a much more intuitive representation of the angular and linear velocities than when we use transformation matrices.

The rigid body spatial velocity V^s_{xy} is a vector comprised of the instantaneous spatial angular velocity w^s_{xy} and the linear component v^s_{xy}:

$$V^s_{xy} = \begin{bmatrix} w^s_{xy} \\ v^s_{xy} \end{bmatrix}. \tag{5.49}$$

The tangential velocity of a point p_x attached to the rigid body measured with respect to the spatial frame X is given by

$$v^s_{p_x} = \widehat{V}^s_{xy} \begin{bmatrix} p_x \\ 1 \end{bmatrix} = \hat{w}^s_{xy}p_x + v^s_{xy} = w^s_{xy} \times p_x + v^s_{xy}. \tag{5.50}$$

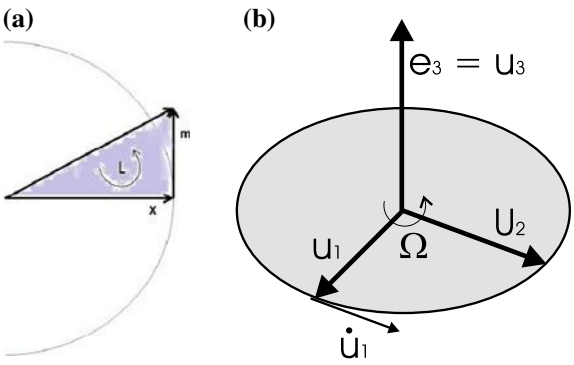


Fig. 5.4 **a** (left) The particle sweeps out the plane $\boldsymbol{L} = \boldsymbol{x} \wedge \boldsymbol{m}$. **b** (right) Rotating frame $\{e_k\}$ with $\boldsymbol{R}$. Bivector angular velocity $\boldsymbol{\Omega} = \boldsymbol{u}_1 \wedge \dot{\boldsymbol{u}}_1$, if the vector $\boldsymbol{\omega}$ has the same orientation as $e_1 \wedge e_2$ then $\boldsymbol{\omega} = |\boldsymbol{\omega}|e_3$

After this review of well-known motion formulas using matrices, we will represent the rigid body spatial velocity now using the more advantageous techniques of motor algebra.

5.8.2 *Angular Velocity Using Rotors*

The angular momentum of a particle with momentum $\boldsymbol{m}$ and position vector $\boldsymbol{x}$ is usually defined in 3D by using the cross product

$$L = \boldsymbol{x} \times \boldsymbol{m}. \tag{5.51}$$

In geometric algebra, one replaces axial vectors with bivectors: thus, we rewrite the last equation using a bivector:

$$\boldsymbol{L} = \boldsymbol{x} \wedge \boldsymbol{m}, \tag{5.52}$$

see this formulation in the domain of physics in [2]. This formula substitutes the old notion of angular momentum as "an axial vector" with a geometric expression that describes the angular momentum as a particle sweeping out a plane (see Fig. 5.4a).

Since angular momentum is described as a bivector, the angular velocity must be represented as a bivector as well. To do so, we resort to a rotor equation. Suppose the orthonormal frame $\{u_k\}$ is rotating in the 3D space, and it is related to another via a rotor $\boldsymbol{R}$:

$$u_k = \boldsymbol{R}(t) e_k \tilde{\boldsymbol{R}}(t). \tag{5.53}$$

Traditionally, the angular momentum vector $\boldsymbol{w}$ is defined using the cross product

$$\dot{u}_k = \boldsymbol{w} \times u_k = -I\boldsymbol{w} \wedge u_k = (-I\boldsymbol{w}) \cdot u_k. \tag{5.54}$$

In this equation, the (space) angular-velocity bivector is

$$\boldsymbol{\Omega}_S = I_3 \boldsymbol{w}, \tag{5.55}$$

where the pseudoscalar $I_3 \in G_{3,0,0}$. The sign ensures the orientation sense followed by the involved rotor. Now, let us analyze the time dependence with respect to the frame $\{u_k\}$:

$$\dot{u} = \dot{\boldsymbol{R}} e_k \tilde{\boldsymbol{R}} + \boldsymbol{R} e_k \dot{\tilde{\boldsymbol{R}}} = \dot{\boldsymbol{R}} \tilde{\boldsymbol{R}} u_k + u_k \boldsymbol{R} \dot{\tilde{\boldsymbol{R}}}. \tag{5.56}$$

Since $\boldsymbol{R}\tilde{\boldsymbol{R}} = 1$, we derive

$$\partial_t (\boldsymbol{R}\tilde{\boldsymbol{R}}) = \dot{\boldsymbol{R}}\tilde{\boldsymbol{R}} + \boldsymbol{R}\dot{\tilde{\boldsymbol{R}}} = 0, \tag{5.57}$$

which leads to

$$\dot{\boldsymbol{R}}\tilde{\boldsymbol{R}} = -\boldsymbol{R}\dot{\tilde{\boldsymbol{R}}}. \tag{5.58}$$

We substitute Eq. (5.58) into Eq. (5.56): the result is simply the inner product of a bivector with a vector:

$$\dot{u} = \dot{\boldsymbol{R}}\tilde{\boldsymbol{R}} u_k - u_k \dot{\boldsymbol{R}}\tilde{\boldsymbol{R}} = (2\dot{\boldsymbol{R}}\tilde{\boldsymbol{R}}) \cdot u_k. \tag{5.59}$$

Comparing Eqs. (5.54) and (5.55), we find an expression for the angular-velocity bivector in terms of the rotor:

$$\boldsymbol{\Omega}_S = -2\dot{\boldsymbol{R}}\tilde{\boldsymbol{R}}. \tag{5.60}$$

If we multiply from the right by a rotor and divide by -2, we easily obtain the dynamic equation reduced to a rotor equation

$$\dot{\boldsymbol{R}} = -\frac{1}{2}\boldsymbol{\Omega}_S \boldsymbol{R}, \tag{5.61}$$

or

$$\dot{\tilde{\boldsymbol{R}}} = \frac{1}{2}\tilde{\boldsymbol{R}}\boldsymbol{\Omega}_S. \tag{5.62}$$

The body angular-velocity $\boldsymbol{\Omega}_B$ related to a fixed-space frame is transformed back as follows:

$$\boldsymbol{\Omega}_S = \boldsymbol{R}\boldsymbol{\Omega}_B\tilde{\boldsymbol{R}}. \tag{5.63}$$

Replacing the last equation into Eq. (5.61) gives

$$\dot{\boldsymbol{R}} = -\frac{1}{2}\boldsymbol{R}\boldsymbol{\Omega}_B = -\frac{1}{2}\boldsymbol{\Omega}_S\boldsymbol{R} \tag{5.64}$$

and into Eq. (5.62) gives

$$\dot{\tilde{\boldsymbol{R}}} = \frac{1}{2}\boldsymbol{\Omega}_B\tilde{\boldsymbol{R}}. \tag{5.65}$$

Assuming that the rotor motion is constant through time, such as a body rotating with a fixed angle, then we fix $\boldsymbol{\Omega}_S$ constant and the rotor Eq. (5.64) can be integrated to give

$$\boldsymbol{R}(t) = e^{-\boldsymbol{\Omega}_S\frac{t}{2}}\boldsymbol{R}(0), \tag{5.66}$$

which represents a rotor that rotates with a constant-frequency rotation in the right-hand sense.

5.8.3 Rigid Body Spatial Velocity Using Motor Algebra

In motor algebra, we represent the rigid motion of lines by Eq. (5.17). Lines are represented in terms of two dual bivector bases. Following the same idea followed by the case of the pure rotor motion, we represent the relationship of the moving frames as follows:

$$l'_k(t) = \boldsymbol{M}(t)l_k\widetilde{\boldsymbol{M}}(t), \tag{5.67}$$

where l'_k and l_k are the coefficients of the two frames. Taking its time derivative

$$\dot{l}'_k = \dot{\boldsymbol{M}}l_k\widetilde{\boldsymbol{M}} + \boldsymbol{M}l_k\dot{\widetilde{\boldsymbol{M}}}, \tag{5.68}$$

and substituting $\boldsymbol{M}l_k = l'_k\boldsymbol{M}$, one gets

$$\dot{l}'_k = \dot{\boldsymbol{M}}\widetilde{\boldsymbol{M}}l'_k + l'_k\boldsymbol{M}\dot{\widetilde{\boldsymbol{M}}}. \tag{5.69}$$

Taking the time derivative of the identity $\boldsymbol{M}\widetilde{\boldsymbol{M}} = 1$, we get the useful relations

$$\begin{aligned} \partial_t(\boldsymbol{M}\widetilde{\boldsymbol{M}}) &= 0, \\ \dot{\boldsymbol{M}}\widetilde{\boldsymbol{M}} + \boldsymbol{M}\dot{\widetilde{\boldsymbol{M}}} &= 0, \\ \dot{\boldsymbol{M}}\widetilde{\boldsymbol{M}} &= -\boldsymbol{M}\dot{\widetilde{\boldsymbol{M}}}, \end{aligned} \tag{5.70}$$

which are substituted in Eq. (5.69) to yield

$$\dot{l_k}'(t) = \dot{\boldsymbol{M}}\tilde{\boldsymbol{M}}l'_k - l'_k\dot{\boldsymbol{M}}\tilde{\boldsymbol{M}}. \tag{5.71}$$

Since the right side of the last equation is the inner product between a bivector and the vector basis coefficients, we rewrite it as follows:

$$\dot{l_k}' = (2\dot{\boldsymbol{M}}\tilde{\boldsymbol{M}}) \cdot l'_k. \tag{5.72}$$

Let us call the bivector

$$\boldsymbol{V}_S = 2\dot{\boldsymbol{M}}\tilde{\boldsymbol{M}} \tag{5.73}$$

the spatial velocity; as we will show next it comprises of a bivector for the linear velocity and a bivector for the angular velocity. Recalling that $\boldsymbol{M} = \boldsymbol{T}\boldsymbol{R}$, we proceed as follows

$$\begin{aligned}
\boldsymbol{V}_s &= 2\dot{\boldsymbol{M}}\tilde{\boldsymbol{M}} = 2(\dot{\boldsymbol{T}}\boldsymbol{R} + \boldsymbol{T}\dot{\boldsymbol{R}})\tilde{\boldsymbol{R}}\tilde{\boldsymbol{T}} = 2\dot{\boldsymbol{T}}\boldsymbol{R}\tilde{\boldsymbol{R}}\tilde{\boldsymbol{T}} + 2\boldsymbol{T}\dot{\boldsymbol{R}}\tilde{\boldsymbol{R}}\tilde{\boldsymbol{T}}, \\
&= 2\dot{\boldsymbol{T}}\tilde{\boldsymbol{T}} + 2\Big(1 + I_m\frac{\boldsymbol{t}_s}{2}\Big)\Big(\frac{1}{2}\boldsymbol{\Omega}_s\Big)\Big(1 - I_m\frac{\boldsymbol{t}_s}{2}\Big), \\
&= I_m\dot{\boldsymbol{t}}_s\Big(1 - I\frac{\boldsymbol{t}_s}{2}\Big) + \Big(\boldsymbol{\Omega}_s + I_m\frac{\boldsymbol{t}_s}{2}\boldsymbol{\Omega}_s\Big)\Big(1 - I_m\frac{\boldsymbol{t}_s}{2}\Big), \\
&= I_m\dot{\boldsymbol{t}}_s + \Big(\boldsymbol{\Omega}_s + I_m\frac{\boldsymbol{t}_s}{2}\boldsymbol{\Omega}_s - I_m\boldsymbol{\Omega}_s\frac{\boldsymbol{t}_s}{2}\Big), \\
&= \boldsymbol{\Omega}_s + I_m\left(\dot{\boldsymbol{t}}_s + \frac{\boldsymbol{t}_s}{2}\boldsymbol{\Omega}_s - \boldsymbol{\Omega}_s\frac{\boldsymbol{t}_s}{2}\right), \\
&= \boldsymbol{\Omega}_s + I_m\Big(\dot{\boldsymbol{t}}_s + \frac{\boldsymbol{t}_s}{2}\underline{\times}\boldsymbol{\Omega}_s\Big) = \boldsymbol{\Omega}_s + I_m\Big(\boldsymbol{v}_s + \frac{\boldsymbol{t}_s}{2}\underline{\times}\boldsymbol{\Omega}_s\Big), \\
&= \boldsymbol{\Omega}_s + I\mathcal{V}_s,
\end{aligned} \tag{5.74}$$

where the angular-velocity bivector $\boldsymbol{\Omega}_s$ is the dual of the linear-velocity bivector $\mathcal{V}_s = \boldsymbol{v}_s + \frac{\boldsymbol{t}_s}{2}\underline{\times}\boldsymbol{\Omega}_s$.

Multiplying the Eq. (5.73) from the right by $\boldsymbol{M}$ and dividing by -2, we get the dynamic motor equation

$$\dot{\boldsymbol{M}} = -\frac{1}{2}\boldsymbol{V}_S\boldsymbol{M}. \tag{5.75}$$

Assuming that the screw motion of the body is constant through time, the dynamic motor Eq. (5.75) can be integrated to give

$$\boldsymbol{M}(t) = e^{-\frac{\boldsymbol{V}_S}{2}}\boldsymbol{M}(0) = e^{-\frac{(\boldsymbol{\Omega}_S + I\sqsubseteq_S)}{2}}\boldsymbol{M}(0). \tag{5.76}$$

This represents a motor that rotates with a constant-frequency rotation in the right-hand sense and has a constant linear velocity as well.

Finally, compare the different algebraic treatments to get Eqs. (5.25), (5.39), (5.61) and (5.75) for the analysis of the rigid-body spatial velocity using matrices, rotor, and motor algebras.

5.8.4 Point, Line, and Plane Spatial Velocities Using Motor Algebra

According to Eq. (5.16), the motion of a point is represented as

$$\boldsymbol{X} = 1 + I\boldsymbol{x} = \boldsymbol{M}\boldsymbol{X}_0\tilde{\boldsymbol{M}}, \tag{5.77}$$

where $\tilde{\boldsymbol{M}} = \tilde{\boldsymbol{R}}\boldsymbol{T}$. If we take the time derivative of this equation, we get

$$\begin{aligned}
\dot{\boldsymbol{X}} &= \dot{\boldsymbol{M}}\boldsymbol{X}_0\tilde{\boldsymbol{M}} + \boldsymbol{M}\boldsymbol{X}_0\dot{\tilde{\boldsymbol{M}}}, \\
&= \dot{\boldsymbol{T}}\boldsymbol{R}\boldsymbol{X}_0\tilde{\boldsymbol{R}}\boldsymbol{T} + \boldsymbol{T}\dot{\boldsymbol{R}}\boldsymbol{X}_0\tilde{\boldsymbol{R}}\boldsymbol{T} + \boldsymbol{R}\boldsymbol{T}\boldsymbol{X}_0\dot{\tilde{\boldsymbol{R}}}\boldsymbol{T} + \boldsymbol{R}\boldsymbol{T}\boldsymbol{X}_0\tilde{\boldsymbol{R}}\dot{\boldsymbol{T}}, \\
&= \dot{\boldsymbol{T}}\tilde{\boldsymbol{T}}\boldsymbol{T}\boldsymbol{R}\boldsymbol{X}_0\tilde{\boldsymbol{R}}\boldsymbol{T} + \boldsymbol{T}\dot{\boldsymbol{R}}\tilde{\boldsymbol{R}}\boldsymbol{R}\boldsymbol{X}_0\tilde{\boldsymbol{R}}\boldsymbol{T} + \boldsymbol{R}\boldsymbol{T}\boldsymbol{X}_0\tilde{\boldsymbol{R}}\boldsymbol{R}\dot{\tilde{\boldsymbol{R}}}\boldsymbol{T} + \boldsymbol{R}\boldsymbol{T}\boldsymbol{X}_0\tilde{\boldsymbol{R}}\dot{\boldsymbol{T}}\tilde{\boldsymbol{T}}\boldsymbol{T}, \\
&= \frac{1}{2}\left[\boldsymbol{\Omega}_S\boldsymbol{X}_0 - \boldsymbol{X}_0\boldsymbol{\Omega}_S\right] + I\frac{1}{2}\left[\dot{\boldsymbol{t}}_S\boldsymbol{X}_0 + \boldsymbol{X}_0\dot{\boldsymbol{t}}_S\right] = \boldsymbol{\Omega}_S \wedge \boldsymbol{X}_0 + (I\boldsymbol{v}_S)\cdot\boldsymbol{X}_0, \\
&= I\boldsymbol{w} + (I\boldsymbol{v}_S)\cdot\boldsymbol{X}_0.
\end{aligned} \tag{5.78}$$

According to Eqs. (5.5) and (5.7), the motion equations of a line and a plane are represented as

$$\begin{aligned}
\boldsymbol{L} &= \boldsymbol{n} + I\boldsymbol{m} = \boldsymbol{M}\boldsymbol{L}_0\tilde{\boldsymbol{M}}, \\
\boldsymbol{H} &= \boldsymbol{n} + Id = \boldsymbol{M}\boldsymbol{H}_0\tilde{\boldsymbol{M}}.
\end{aligned} \tag{5.79}$$

If we take the time derivative of these equations and follow simple algebraic equations similar to the motion equation of the point, we get

$$\dot{\boldsymbol{L}} = \boldsymbol{\Omega}_S \wedge \boldsymbol{L}, \tag{5.80}$$

$$\dot{\boldsymbol{H}} = \boldsymbol{\Omega}_S \wedge \boldsymbol{H} + (I\dot{\boldsymbol{t}}_S)\cdot\boldsymbol{H} = \boldsymbol{\Omega}_S \wedge \boldsymbol{H} + (I\boldsymbol{v}_S)\cdot\boldsymbol{n}. \tag{5.81}$$

Let us consider the velocity $\boldsymbol{V}_S$ when a composition of two motors $\boldsymbol{M} = \boldsymbol{M}_2\boldsymbol{M}_1$. According to Eq. (5.73), we get

$$\begin{aligned}
\boldsymbol{V}_S &= -2\dot{\boldsymbol{M}}\tilde{\boldsymbol{M}} = -2(\dot{\boldsymbol{M}}_2\boldsymbol{M}_1 + \boldsymbol{M}_2\dot{\boldsymbol{M}}_1)\widetilde{(\boldsymbol{M}_2\boldsymbol{M}_1)}, \\
&= -2(\dot{\boldsymbol{M}}_2\boldsymbol{M}_1 + \boldsymbol{M}_2\dot{\boldsymbol{M}}_1)\tilde{\boldsymbol{M}}_1\tilde{\boldsymbol{M}}_2 = -2\dot{\boldsymbol{M}}_2\tilde{\boldsymbol{M}}_2 + \boldsymbol{M}_2(-2\dot{\boldsymbol{M}}_1\tilde{\boldsymbol{M}}_1)\tilde{\boldsymbol{M}}_2, \\
&= \boldsymbol{V}_2 + \boldsymbol{M}_2\boldsymbol{V}_1\tilde{\boldsymbol{M}}_2.
\end{aligned} \tag{5.82}$$

In general, for a sequence of n motors, the overall velocity $\boldsymbol{V}_S$ is given as follows:

$$\begin{aligned}\boldsymbol{V}_S = \; & \boldsymbol{V}_n + \boldsymbol{M}_n\boldsymbol{M}_{n-1}\boldsymbol{M}_{n-2}\cdots\boldsymbol{M}_{n-1}\boldsymbol{V}_{n-1}\tilde{\boldsymbol{M}}_{n-1}\cdots\tilde{\boldsymbol{M}}_{n-1}\tilde{\boldsymbol{M}}_n + \\ & \boldsymbol{M}_{n-1}\boldsymbol{M}_{n-2}\boldsymbol{M}_{n-3}\cdots\boldsymbol{M}_{n-1}\boldsymbol{V}_{n-2}\tilde{\boldsymbol{M}}_{n-1}\cdots\tilde{\boldsymbol{M}}_{n-3}\tilde{\boldsymbol{M}}_{n-2}\tilde{\boldsymbol{M}}_{n-1} + \\ & \cdots + \boldsymbol{M}_3\boldsymbol{M}_2\boldsymbol{V}_2\tilde{\boldsymbol{M}}_3\tilde{\boldsymbol{M}}_2 + \boldsymbol{M}_2\boldsymbol{V}_1\tilde{\boldsymbol{M}}_2, \\ = \; & \sum_{i=1}^{n}\prod_{j=i+1}^{n}\boldsymbol{M}_j\boldsymbol{V}_i\tilde{\boldsymbol{M}}_j. \end{aligned} \tag{5.83}$$

5.9 Differential Kinematics

In the conformal geometric algebra framework, the direct kinematics for serial robot arms is a succession of motors and is valid for points, lines, planes, circles, and spheres:

$$\boldsymbol{Q}' = \prod_{i=1}^{n}\boldsymbol{M}_i\,\boldsymbol{Q}\prod_{i=1}^{n}\widetilde{\boldsymbol{M}}_{n-i+1}. \tag{5.84}$$

The direct kinematics Eq. (5.84) can be used for points as

$$\boldsymbol{x}'_p = \prod_{i=1}^{n}\boldsymbol{M}_i\boldsymbol{x}_p\prod_{i=1}^{n}\widetilde{\boldsymbol{M}}_{n-i+1}. \tag{5.85}$$

This equation could be used in conformal geometric algebra, using motors to represent 3D rigid transformations similar as with the motor algebra (see Sect. 4.5). Now we produce an expression for differential kinematics via the total differentiation of (5.85):

$$d\boldsymbol{x}'_p = \sum_{j=1}^{n}\partial_{q_j}\left(\prod_{i=1}^{n}\boldsymbol{M}_i\boldsymbol{x}_p\prod_{i=1}^{n}\widetilde{\boldsymbol{M}}_{n-i+1}\right)dq_j. \tag{5.86}$$

Each term of the sum is the product of two functions in q_j, and the differential can be written as

$$\begin{aligned} d\boldsymbol{x}'_p = \sum_{j=1}^{n}\Bigg[& \partial_{q_j}\left(\prod_{i=1}^{j}\boldsymbol{M}_i\right)\prod_{i=j+1}^{n}\boldsymbol{M}_i\boldsymbol{x}_p\prod_{i=1}^{n}\widetilde{\boldsymbol{M}}_{n-i+1} \\ & +\prod_{i=1}^{n}\boldsymbol{M}_i\boldsymbol{x}_o\prod_{i=1}^{n-j}\widetilde{\boldsymbol{M}}_{n-i+1}\partial_{q_j}\left(\prod_{i=n-j+1}^{n}\widetilde{\boldsymbol{M}}_{n-i+1}\right)\Bigg]dq_j. \end{aligned} \tag{5.87}$$

Since $\boldsymbol{M} = e^{-\frac{1}{2}q\boldsymbol{L}}$, the differential of the motor is $d(\boldsymbol{M}) = -\frac{1}{2}\boldsymbol{M}\boldsymbol{L}dq$; thus, we can write the partial differential of the motor's product as follows:

$$\partial_{q_j}\left(\prod_{i=1}^{j}\boldsymbol{M}_i\right) = -\frac{1}{2}\prod_{i=1}^{j}\boldsymbol{M}_i\boldsymbol{L}_j = -\frac{1}{2}\left(\prod_{i=1}^{j-1}\boldsymbol{M}_i\right)\boldsymbol{L}_j\boldsymbol{M}_j. \tag{5.88}$$

Similarly, the differential of $\widetilde{\boldsymbol{M}} = e^{\frac{1}{2}q\boldsymbol{L}}$ gives us $d(\widetilde{\boldsymbol{M}}) = \frac{1}{2}\boldsymbol{M}Ldq$, and the differential of the product is

$$\partial_{q_j}\left(\prod_{i=n-j+1}^{n}\widetilde{\boldsymbol{M}}_{n-i+1}\right) = \frac{1}{2}\widetilde{\boldsymbol{M}}_j\boldsymbol{L}_j\prod_{i=n-j+2}^{n}\widetilde{\boldsymbol{M}}_{n-i+1}. \tag{5.89}$$

Replacing (5.88) and (5.89) in (5.87), we get

$$\begin{aligned} d\boldsymbol{x}'_p = \sum_{j=1}^{n}\Bigg[& -\frac{1}{2}\prod_{i=1}^{j-1}\boldsymbol{M}_i\boldsymbol{L}_j\boldsymbol{M}_j\prod_{i=j+1}^{n}\boldsymbol{M}_i\boldsymbol{x}_p\prod_{i=1}^{n}\widetilde{\boldsymbol{M}}_{n-i+1} \\ & +\frac{1}{2}\prod_{i=1}^{n}\boldsymbol{M}_i\boldsymbol{x}_o\prod_{i=1}^{n-j}\widetilde{\boldsymbol{M}}_{n-i+1}\widetilde{\boldsymbol{M}}_j\boldsymbol{L}_j\prod_{i=1}^{j-1}\widetilde{\boldsymbol{M}}_{j-i}\Bigg]dq_j, \end{aligned} \tag{5.90}$$

which can be further simplified as

$$\begin{aligned} d\boldsymbol{x}'_p = -\frac{1}{2}\sum_{j=1}^{n}\Bigg[& \prod_{i=1}^{j-1}\boldsymbol{M}_i\left(\boldsymbol{L}_j\left(\prod_{i=j}^{n}\boldsymbol{M}_i x_o\prod_{i=1}^{n-j+1}\widetilde{\boldsymbol{M}}_{n-i+1}\right)\right. \\ & \left.-\left(\prod_{i=j}^{n}\boldsymbol{M}_i x_p\prod_{i=1}^{n-j+1}\widetilde{\boldsymbol{M}}_{n-i+1}\right)\boldsymbol{L}_j\right)\prod_{i=1}^{j-1}\widetilde{\boldsymbol{M}}_{j-i}\Bigg]dq_j. \end{aligned} \tag{5.91}$$

Note that the product of a vector with an r-vector is given by (see [3])

$$a\cdot B_r = \frac{1}{2}\left(aB_r + (-1)^{r+1}B_r a\right). \tag{5.92}$$

Using Eq. (5.92), we can further simplify (5.91). As L is a bivector and x_p is a vector, we will rewrite (5.91) as follows:

$$d\boldsymbol{x}'_p = \sum_{j=1}^{n}\left[\left(\prod_{i=1}^{j-1}\boldsymbol{M}_i\right)\left(\left(\prod_{i=j}^{n}\boldsymbol{M}_i\boldsymbol{x}_p\prod_{i=1}^{n-j+1}\widetilde{\boldsymbol{M}}_{n-i+1}\right)\cdot\boldsymbol{L}_j\right)\cdot\left(\prod_{i=1}^{j-1}\widetilde{\boldsymbol{M}}_{j-i}\right)\right]dq_j. \tag{5.93}$$

As with points, all the transformations in conformal geometric algebra can also be applied to the lines; thus,

$$d\boldsymbol{x}'_p = \sum_{j=1}^{n}\left[\left(\prod_{i=1}^{j-1}\boldsymbol{M}_i\prod_{i=j}^{n}\boldsymbol{M}_i\boldsymbol{x}_p\prod_{i=1}^{n-j+1}\widetilde{\boldsymbol{M}}_{n-i+1}\prod_{i=1}^{j-1}\widetilde{\boldsymbol{M}}_{j-i}\right)\cdot\left(\prod_{i=1}^{j-1}\boldsymbol{M}_i\boldsymbol{L}_j\prod_{i=1}^{j-1}\widetilde{\boldsymbol{M}}_{j-i}\right)\right]dq_j. \tag{5.94}$$

Since $\prod_{i=1}^{j-1}\boldsymbol{M}_i\prod_{i=j}^{n}\boldsymbol{M}_i = \prod_{i=1}^{n}\boldsymbol{M}_i$, we have

$$d\boldsymbol{x}'_p = \sum_{j=1}^{n}\left[\left(\prod_{i=1}^{n}\boldsymbol{M}_i\boldsymbol{x}_p\prod_{i=1}^{n}\widetilde{\boldsymbol{M}}_{n-i+1}\right)\left(\prod_{i=1}^{j-1}\boldsymbol{M}_i\boldsymbol{L}_j\prod_{i=1}^{j-1}\widetilde{\boldsymbol{M}}_{j-i}\right)\right]dq_j. \tag{5.95}$$

Recall Eq. (5.85) of the direct kinematics; since in Eq. (5.95) $\boldsymbol{x}'_p$ appears again, we can replace (5.85) in (5.95) to get

$$d\boldsymbol{x}'_p = \sum_{j=1}^{n}\left[\boldsymbol{x}'_p\cdot\left(\prod_{i=1}^{j-1}\boldsymbol{M}_i\boldsymbol{L}_j\prod_{i=1}^{j-1}\widetilde{\boldsymbol{M}}_{j-i}\right)\right]dq_j. \tag{5.96}$$

If we define L' as function of L as follows:

$$\boldsymbol{L}'_j = \prod_{i=1}^{j-1}\boldsymbol{M}_i\boldsymbol{L}_j\prod_{i=1}^{j-1}\widetilde{\boldsymbol{M}}_{j-i}, \tag{5.97}$$

we get a very compact expression of differential kinematics:

$$d\boldsymbol{x}'_p = \sum_{j=1}^{n}\left[\boldsymbol{x}'_p\cdot\boldsymbol{L}'_j\right]dq_j. \tag{5.98}$$

In this way, we can finally write

$$\dot{\boldsymbol{x}}'_p = \left(\boldsymbol{x}'_p\cdot\boldsymbol{L}'_1\cdots\boldsymbol{x}'_p\cdot\boldsymbol{L}'_n\right)\begin{pmatrix}\dot{q}_1\\ \vdots\\ \dot{q}_n.\end{pmatrix}. \tag{5.99}$$

5.10 Incidence Relations Between Points, Lines, and Planes

The geometric relations between points, lines, and planes expressed in terms of incidence relations are very useful when we are dealing with the geometry of configurations and the relative motion of objects. Blaske introduced the basic relations of incidence using dual quaternions [4]. These relations can be also formulated for the representations of points, lines, and planes in the motor algebra $\mathcal{G}^{+}_{3,0,1}$ or in the 4D degenerated geometric algebra $G_{3,0,1}$. In general, the incidence relations between a point, Line, and a plane are given by

$$\widetilde{\boldsymbol{P}}\boldsymbol{L} + \widetilde{\boldsymbol{L}}\boldsymbol{P} = -2I(\boldsymbol{x} \cdot \boldsymbol{n}), \tag{5.100}$$

$$\widetilde{\boldsymbol{P}}\boldsymbol{\Pi} + \widetilde{\boldsymbol{\Pi}}\boldsymbol{P} = -2I(d - \boldsymbol{x} \cdot \boldsymbol{n}_{\pi}), \tag{5.101}$$

$$\widetilde{\boldsymbol{L}}\boldsymbol{\Pi} + \widetilde{\boldsymbol{\Pi}}\boldsymbol{L} = -2(\boldsymbol{n}_{L} \cdot \boldsymbol{n}_{\pi} + \boldsymbol{m}_{L} \cdot \boldsymbol{n}_{\pi}), \tag{5.102}$$

where $\widetilde{\boldsymbol{P}}$ and $\widetilde{\boldsymbol{L}}$ are their conjugated (reversion), respectively.

Thus, a point $\boldsymbol{P}$ lying on a line $\boldsymbol{L}$ fulfills the equation

$$\widetilde{\boldsymbol{P}}\boldsymbol{L} + \widetilde{\boldsymbol{L}}\boldsymbol{P} = 0. \tag{5.103}$$

The distance d of a point relative to a plane Π is given by

$$\widetilde{\boldsymbol{P}}\boldsymbol{\Pi} + \widetilde{\boldsymbol{\Pi}}\boldsymbol{P} = de_1e_2e_3e_4. \tag{5.104}$$

If $d = 0$, the point lies on the plane; if d is negative, it is behind the plane; and if d is positive, it is in front of the plane.

The intersection point $\boldsymbol{P}$ of a line $\boldsymbol{L}$ crossing a plane Π can be computed as

$$\widetilde{\boldsymbol{L}}\boldsymbol{\Pi} + \widetilde{\boldsymbol{\Pi}}\boldsymbol{L} = \boldsymbol{P}. \tag{5.105}$$

In this case, if the line is parallel to the plane, the equation equals zero.

Incidence relations are fundamental in projective geometry, and we will study in more detail in next chapters.

5.10.1 Flags of Points, Lines, and Planes

When dealing with geometric configurations as in object modeling or robot navigation planning, it is useful to resort to a type of *geometric indicator*, which allows us to detect whether a geometric condition is fulfilled or not. These kinds of indicators are called flags, and they are expressions relating points, lines, and planes that have some common attributes. Flags generate varieties [5].

We can express a point touching a line as the equation of the so-called *point-line flag*:

$$\boldsymbol{\mathcal{F}}_{PL} = \boldsymbol{P} + \boldsymbol{L}. \tag{5.106}$$

If $\boldsymbol{P}\widetilde{\boldsymbol{P}} = 1$ and $\boldsymbol{L}\widetilde{\boldsymbol{L}} = 1$, then $\mathcal{F}_{PL}\mathcal{F}^{*}_{PL} = 1$.

If we have a point which touches a line and also a plane, and in which the orientation of the plane is parallel to the line, then we can represent the *line–plane flag* as

$$\boldsymbol{\mathcal{F}}_{L\Pi} = \boldsymbol{L} + \boldsymbol{\Pi}. \tag{5.107}$$

In this equation, if $\boldsymbol{L}\widetilde{\boldsymbol{L}} = 1$ and $\boldsymbol{\Pi}\widetilde{\boldsymbol{\Pi}} = 1$, then $\mathcal{F}_{L\Pi}\widetilde{\mathcal{F}}_{L\Pi} = 1$.

Finally, if a point touches a line and a plane and the line has the same orientation as the normal to the plane, we can assign to this geometry the following *point–line–plane flag*:

$$\boldsymbol{\mathcal{F}}_{PL\Pi} = \boldsymbol{P} + \boldsymbol{L} + \boldsymbol{\Pi}. \tag{5.108}$$

Here, if $\boldsymbol{P}\widetilde{\boldsymbol{P}} = 1$, $\boldsymbol{L}\widetilde{\boldsymbol{L}} = 1$, and $\boldsymbol{\Pi}\widetilde{\boldsymbol{\Pi}} = 1$, then $\mathcal{F}_{PL\Pi}\widetilde{\mathcal{F}}_{PL\Pi} = 1$.

5.11 Conclusion

This chapter presents the Clifford or geometric algebra for computations in visually guided robotics. Looking for other suitable ways of representing algebraic relations of geometric primitives, we consider the complex and dual numbers in the geometric algebra framework. It turns out that in this framework the algebra of motors is well suited to express the 3D kinematics. Doing that we can linearize the nonlinear 3D rigid motion transformation. In this chapter, the geometric primitives' points, lines, and planes are represented using the 3D Euclidean geometric algebra and the 4D motor algebra. Next, the rigid motions of these geometric primitives are elegantly expressed using rotors, motors, and concepts of duality. In the algebra of motors, we extend the 3D Euclidean space representation to a 4D space by means of a dual copy of scalars, vectors, and rotors or quaternions. Finally, we formulate incidence relations between points, lines, and planes using the motor algebra framework.

5.12 Exercises

5.1 Split the vector $\boldsymbol{v} = -2e_1 + 7e_2 + 10e_3$ into the components $\boldsymbol{x} = 3e_1 - 1.5e_2$, $\boldsymbol{y} = 2e_1 + 4e_2 - 2e_3$, and $\boldsymbol{z} = 5e_1 + 3e_2 + 3e_3$. In other words, compute the coefficients $\alpha,\ \beta,\ e$ of the equation $\boldsymbol{v} = \alpha\boldsymbol{x} + \beta\boldsymbol{y} + e\boldsymbol{z}$. Draw in three dimensions the vectors and the volume $\boldsymbol{a} \wedge \boldsymbol{b} \wedge \boldsymbol{c}$.

5.2 The outer square root of a multivector $\boldsymbol{M}$ is $\boldsymbol{x}$, which is the solution of the equation $\boldsymbol{M} = \boldsymbol{x} \wedge \boldsymbol{x}$. If $\boldsymbol{M} = \alpha + \boldsymbol{a}$, where $\alpha \in R$, we can write $\boldsymbol{x} =$

$\sqrt{(}\alpha)(1+\frac{\boldsymbol{a}}{\alpha})^{\frac{1}{2}}$. This equation can be expanded using the following series:

$$\left(1+\frac{\boldsymbol{a}}{\alpha}\right)^{\frac{1}{2}} = 1+\frac{1}{2}\frac{\boldsymbol{a}}{\alpha}+\frac{\frac{1}{2}(\frac{1}{2}-1)}{2}\cdot\frac{\boldsymbol{a}\wedge\boldsymbol{a}}{\alpha^2}+\frac{\frac{1}{2}(\frac{1}{2}-1)(\frac{1}{2}-2)}{3!}\cdot\frac{\boldsymbol{a}\wedge\boldsymbol{a}\wedge\boldsymbol{a}}{\alpha^3}+\cdots$$

The quantity of the elements of this series should be at least equal to the dimension of the considered geometric algebra. Generate a program file for CLICAL [6] or eClifford [7] to compute the outer square root of $\boldsymbol{M}$, where $\mathrm{Re}(\boldsymbol{M}) > 0$. Give any $\boldsymbol{M}$ and verify the identity $\boldsymbol{x}\wedge\boldsymbol{x} = \boldsymbol{M}$.

5.3 Rotate in 4D Euclidean space, using CLICAL [6] or eClifford [7], the vector $\boldsymbol{v} = e_1 + e_2 + e_3$, first related to the plane e_1e_2 by the angle $\frac{\phi}{4}$, and then related to the plane e_4e_2 by the angle $\frac{\phi}{5}$.

5.4 Given $\boldsymbol{x} = x_1e_1 + x_2e_2 + x_3e_3$, $\boldsymbol{y} = y_1e_1 + y_2e_2 + y_3e_3$, $\boldsymbol{X} = \boldsymbol{x}e_1e_2e_3$, and $\boldsymbol{Y} = \boldsymbol{y}e_1e_2e_3$, compute the bivectors $\frac{1}{2}(1+\epsilon_1e_2e_3e_4)\boldsymbol{X}$ and $\frac{1}{2}(1-e_1e_2e_3e_4)\boldsymbol{Y}$ and show that they commute.

5.5 Given the bivector $\boldsymbol{B} = \alpha e_1e_2 + \beta e_4e_3$, compute $\boldsymbol{B}\cdot\boldsymbol{B}$, $\boldsymbol{B}\wedge\boldsymbol{B}$, and $\boldsymbol{B}\times\boldsymbol{B}$ and explain what kinds of multivectors result.

5.6 For this problem, use the same multivectors $\boldsymbol{X}$ and $\boldsymbol{Y}$ that you used in Exercise 5.4. Given $\boldsymbol{Z} = \frac{1}{2}(1+e_1e_2e_3e_4)\boldsymbol{X} + \frac{1}{2}(1-e_1e_2e_3e_4)\boldsymbol{Y}$, express $\exp(\boldsymbol{Z})$ using $|\boldsymbol{x}|$ and $|\boldsymbol{y}|$. What are the rotation angles of the rotation $\mathbb{R}^4\to\mathbb{R}^4$, $\boldsymbol{u}\to\boldsymbol{z}\boldsymbol{u}\boldsymbol{z}^{-1}$, where $\boldsymbol{z} = exp(\boldsymbol{Z})$?

5.7 Given in $G_{3,0,0}$ the points $\boldsymbol{a} = 0.5e_1 + 2.0e_2 + 1.3e_3$, $\boldsymbol{b} = 1.5e_1 + 1.2e_2 + 2.3e_3$, and $\boldsymbol{c} = 0.7e_1 + 1.2e_2 - 0.3e_3$, compute the line $\boldsymbol{l}$ crossing $\boldsymbol{a}$ and $\boldsymbol{b}$, and the plane ϕ tangent to the points $\boldsymbol{a}$, $\boldsymbol{b}$, and $\boldsymbol{c}$.

5.8 Using the software packet eClifford [7] and point $\boldsymbol{a}$, line $\boldsymbol{l}$, and plane $\boldsymbol{\phi}$ from Exercise 5.7, in $G_{3,0,0}$ compute new values for the point, line, and plane after undergoing a rigid motion given by the translation $\boldsymbol{t} = 1.0e_1 - 2.7e_2 + 5.3e_3$. Also compute the rotor $\boldsymbol{R} = \cos(\frac{\theta}{2}) + \sin(\frac{\theta}{2})\boldsymbol{n}$, where $\theta = \frac{\pi}{6}$ and $\boldsymbol{n}$ is the unit 3D bivector of the rotation axis given by $\boldsymbol{n} = 0.7e_2e_3 + 1.2e_3e_1 + 0.9e_1e_2$.

5.9 Express point $\boldsymbol{a}$, line $\boldsymbol{l}$, and plane $\boldsymbol{\phi}$ given in Exercise 5.7 in the algebra of motors $G^+_{3,0,1}$.

5.10 Using the software packet eClifford and the point, line, and plane given in Exercise 5.7, compute their new values after undergoing the rigid motion given by the translator $\boldsymbol{T} = 1 + I\frac{1.0\gamma_2\gamma_3 - 2.7\gamma_3\gamma_1 + 5.3\gamma_1\gamma_2}{2}$ and the rotor $\boldsymbol{R}_s = \cos(\frac{\theta}{2}) + \sin(\frac{\theta}{2})\boldsymbol{l}$, where $\theta = \frac{\pi}{6}$ and $\boldsymbol{l}$ is the screw-axis line. Compare your results with the results of Exercise 5.8.

5.11 According to c.f. [8, Exercise 3.3], to solve this exercise consider in Sect. 5.8.2 Eq. (5.66).

$$\boldsymbol{R}(t) = e^{-\boldsymbol{\Omega}_S\frac{t}{2}}\boldsymbol{R}(0).$$

Let be a particle in the 3-D space which moves along a curve $\boldsymbol{c}(t)$ with constant velocity $|\boldsymbol{v}|$, prove that there exists a bivector Ω such that

$$\dot{\boldsymbol{v}} = \Omega \cdot \boldsymbol{v}.$$

Give an explicit formula for the bivector Ω. Is this bivector unique?

5.12 According to c.f. [8, Exercise 3.4] and Sect. 5.8.2, imagine you measure the components of the position vector $\boldsymbol{x}$ in a rotating frame $\{f_i\}$. Referring this frame to a fixed frame, prove that the components of $\boldsymbol{x}$ are given by

$$x_i = e_i \cdot (\boldsymbol{R}\boldsymbol{x}\widetilde{\boldsymbol{R}}). \tag{5.109}$$

Now, differentiate this expression twice and prove that one can write

$$f_i \ddot{x}_i = \ddot{\boldsymbol{x}} + \Omega \cdot (\Omega \cdot \boldsymbol{x}) + 2\Omega \cdot \dot{\boldsymbol{x}} + \dot{\Omega} \cdot \boldsymbol{x}.$$

Using this, deduce the expressions for the centrifugal, Coriolis, and Euler forces in terms of the angular-velocity bivector Ω.

5.13 Express the geometric objects of 5.7 in the degenerated algebra $G_{3,0,1}$: the point $\boldsymbol{a}$ using trivectors, the line $\boldsymbol{l}$ using bivectors, and the plane $\boldsymbol{\phi}$ using vectors.

5.14 Using the software packet eClifford and point $\boldsymbol{a}$, line $\mathbf{1}$, and plane ϕ from Exercise 5.11, compute new values for the point, line, and plane after undergoing the rigid motion given by the translator $\boldsymbol{T} = 1 + I\frac{1.0\gamma_2\gamma_3 - 2.7\gamma_3\gamma_1 + 5.3\gamma_1\gamma_2}{2}$. Also, compute the rotor $\boldsymbol{R}_s = \cos(\frac{\theta}{2}) + \sin(\frac{\theta}{2})\boldsymbol{l}$, where $\theta = \frac{\pi}{6}$ and $\boldsymbol{l}$ is the screw-axis line. Compare your results with the results of Exercises 5.8 and 5.10.

5.15 Prove the Rodriguez formula for 3D rigid motion using motor algebra. Compare your results with the results from Exercise 4.2. Why is the motor algebra expression superior?

5.16 In the degenerated algebra $G_{3,0,1}$, choose a point $\boldsymbol{P}$ using trivectors and a line $\boldsymbol{L}$ using bivectors, and prove that the following expression,

$$\widetilde{\boldsymbol{P}}\mathbf{L} + \widetilde{\boldsymbol{L}}\boldsymbol{P},$$

equals zero if the point lies on the line. Prove the same equation using representations of the point and line in the motor algebra $G^{+}_{3,0,1}$.

5.17 In the degenerated algebra $G_{3,0,1}$ for the point $\boldsymbol{P}$ using trivectors, and the plane $\boldsymbol{\Pi}$ using vectors, show that the following expression,

$$\widetilde{\boldsymbol{P}}\boldsymbol{\Pi} + \widetilde{\boldsymbol{\Pi}}\boldsymbol{P} = de_1e_2e_3e_4,$$

will describe a particular geometric configuration, depending upon the value of d. If d=0, the point lies on the plane; if $d \neq 0$, d indicates the distance of the point to the plane; and if d is negative, the point is behind the plane. Using eClifford [7] and some points of the 3D space, check this equation. Check the

equation again using the representations of the point and plane in the motor algebra $G^{+}_{3,0,1}$.

5.18 In the degenerated algebra $G_{3,0,1}$ for the line $\boldsymbol{L}$ using bivectors, the plane $\boldsymbol{\Pi}$ using vectors shows that the following expression,

$$\widetilde{\boldsymbol{L}}\boldsymbol{\Pi} + \widetilde{\boldsymbol{L}}\boldsymbol{\Pi},$$

will describe a particular geometric configuration. If this expression equals zero, the line lies on the plane or is parallel to it; if not, the equation yields the intersecting point. This equation can be seen as a kind of meet equation when using a degenerated algebra. Using eClifford and some points of 3D space, check the equation. Check the same equation using the representations of the line and plane in the motor algebra $G^{+}_{3,0,1}$.

5.19 Three points $\boldsymbol{x}$, $\boldsymbol{y}$, $\boldsymbol{z}$ are in general position lying on a 2D simplex (a plane). They can be used to describe any point $\boldsymbol{p}$ on this plane:

$$\boldsymbol{p} = s_1\boldsymbol{x} + s_2\boldsymbol{y} + s_3\boldsymbol{z},$$

where the scalars s_i are known as the barycentric coordinates. Compute the scalars s_i in terms of ratios of bivectors. Interpret your results. Are the scalars s_i coordinates of the gravity center?

5.20 Construct the dual representation of the mid-plane between points $\boldsymbol{x}$ and $\boldsymbol{y}$.

5.21 Flags: Flags are very useful when we are interested in relating geometrical points, lines, and planes. Show in the degenerated algebra $G_{3,0,1}$ that the plane perpendicular to the line $\boldsymbol{L}$ passing through the point $\boldsymbol{P}$ is given by

$$\Pi^{\perp} = \frac{1}{2}(\boldsymbol{P}\widetilde{\boldsymbol{L}} - \boldsymbol{L}\widetilde{\boldsymbol{P}}).$$

This equation is a kind of extension which relates the mapping of the flags

$$f_{PL} = \frac{1}{\sqrt{2}}(\boldsymbol{P} + \boldsymbol{L}) \rightarrow f_{P\Pi} = \frac{1}{\sqrt{2}}(\boldsymbol{P} + \boldsymbol{\Pi}^{\perp}).$$

The inverse of this mapping relates the plane to the line perpendicular to the plane passing through the point. That is,

$$\boldsymbol{L}^{\perp} = \frac{1}{2}(\boldsymbol{\Pi}\widetilde{\boldsymbol{P}} - \boldsymbol{P}\widetilde{\boldsymbol{\Pi}}).$$

Check that $(\boldsymbol{L}^{\perp})^{\perp} = \boldsymbol{L}$.

References

1. Bayro-Corrochano, E., Daniilidis, K., & Sommer, G. (1997). Hand-eye calibration in terms of motions of lines using geometric algebra. In *10th Scandinavian Conference on Image Analysis* (Vol. I, pp. 397–404). Finland: Lappeenranta.
2. Doran, C. (1994). *Geometric algebra and its applications to mathematical physics*. Ph.D. thesis, University of Cambridge.
3. Hestenes, D., & Sobczyk, G. (1984). *Clifford algebra to geometric calculus: A unified language for mathematics and physics*. Dordrecht: D. Reidel.
4. Blaschke, W. (1960). *Kinematik und Quaternionen*. Berlin: VEB Deutscher Verlag der Wissenschaften.
5. Selig, J. M. (1999). Clifford algebra of points, lines and planes. Technical report SBU-CISM-99-06, South Bank University, School of Computing, Information Technology and Maths.
6. Lounesto, P. (1987). CLICAL software packet and user manual. Helsinki University of Technology of Mathematics, Research report A248.
7. Ablamowicks, R. eCLIFFORD Software packet using Maple for Clifford algebra. Computations. http://math.tntech.edu/rafal.
8. Doran, C., & Lasenby, A. (2005). *Geometric algebra for physicists*. Cambridge University Press.

Chapter 6
Conformal Geometric Algebra

Spacetime tells matter how to move; matter tells spacetime how to curve. John Archibald Wheeler (July 9, 1911 – April 13, 2008)

6.1 Introduction

The geometric algebra of a 3D Euclidean space $G_{3,0,0}$ has a point basis and the motor algebra $G_{3,0,1}$ a line basis. In the latter geometric algebra, the lines expressed in terms of Plücker coordinates can be used to represent points and planes as well. The reader can find a comparison of representations of points, lines, and planes using $G_{3,0,0}$ and $G_{3,0,1}$ in Chap. 5.

Interestingly enough, in the case of the conformal geometric algebra, we find that the unit element is the sphere, which allows us to represent the other geometric primitives in its terms. To see how this is possible, we begin by giving an introduction in conformal geometric algebra following the same formulation presented in [1, 2] and show how the Euclidean vector space $\mathbb{R}^n$ is represented in $\mathbb{R}^{n+1,1}$. Let $\{e_1, \ldots, e_n, e_+, e_-\}$ be a vector basis with the following properties:

$$e_i^2 = 1, \quad i = 1, \ldots, n, \tag{6.1}$$

$$e_\pm^2 = \pm 1, \tag{6.2}$$

$$e_i \cdot e_+ = e_i \cdot e_- = e_+ \cdot e_- = 0, \quad i = 1, \ldots, n. \tag{6.3}$$

Note that this basis is not written in bold. A *null basis* $\{e_0, e_\infty\}$ can be introduced by

$$e_0 = \frac{(e_- - e_+)}{2}, \tag{6.4}$$

$$e_\infty = e_- + e_+, \tag{6.5}$$

© Springer Nature Switzerland AG 2020

E. Bayro-Corrochano, *Geometric Algebra Applications Vol. II*,
https://doi.org/10.1007/978-3-030-34978-3_6

with the properties

$$e_0^2 = e_\infty^2 = 0, \quad e_\infty \cdot e_0 = -1. \tag{6.6}$$

A unit pseudoscalar $\boldsymbol{E} \in \mathbb{R}^{1,1}$ that represents the so-called Minkowski plane is defined by

$$\boldsymbol{E} = e_\infty \wedge e_0 = e_+ \wedge e_- = e_+ e_-, \tag{6.7}$$

having the properties

$$\boldsymbol{E}^2 = 1, \tag{6.8}$$
$$\widetilde{\boldsymbol{E}} = -\boldsymbol{E}, \tag{6.9}$$
$$\boldsymbol{E} e_\pm = e_\mp = -e_\pm, \tag{6.10}$$
$$\boldsymbol{E} e_\infty = -e_\infty \boldsymbol{E} = -e_\infty, \quad \boldsymbol{E} e_0 = -e_0 \boldsymbol{E} = e_0 \quad \text{(absorption)}, \tag{6.11}$$
$$1 - \boldsymbol{E} = -e_\infty e_0, \quad 1 + \boldsymbol{E} = -e_0 e_\infty. \tag{6.12}$$

The dual of $\boldsymbol{E}$ is given by

$$\boldsymbol{E}^* = \boldsymbol{E} I^{-1} = -\boldsymbol{E}\widetilde{I}, \tag{6.13}$$

where I is the pseudoscalar for $\mathbb{R}^{n+1,1}$.

6.1.1 Conformal Split

Euclidean points $\mathbf{x}_e \in \mathbb{R}^n$ can be represented in $\mathbb{R}^{n+1,1}$ in a general way, as

$$\boldsymbol{x}_c = \mathbf{x}_e + \alpha e_0 + \beta e_\infty, \tag{6.14}$$

where α and β are arbitrary scalars. A conformal point $\boldsymbol{x}_c \in \mathbb{R}^{n+1,1}$ can be divided into its Euclidean and conformal parts by an operation called the *additive split*: $\mathbb{R}^{n,1} = \mathbb{R}^n \oplus \mathbb{R}^{1,1}$ [1]. This split is defined by the projection operators P_E (projection) and $P_E^\perp$ (rejection) as follows:

$$P_E(\boldsymbol{x}_c) = (\boldsymbol{x}_c \cdot \boldsymbol{E})\boldsymbol{E} = \alpha e_0 + \beta e_\infty \in \mathbb{R}^{1,1}, \tag{6.15}$$
$$P_E^\perp(\boldsymbol{x}_c) = (\boldsymbol{x}_c \cdot \boldsymbol{E}^*)\widetilde{\boldsymbol{E}^*} = (\boldsymbol{x}_c \wedge \boldsymbol{E})\boldsymbol{E} = \mathbf{x}_e \in \mathbb{R}^n, \tag{6.16}$$
$$\boldsymbol{x}_c = P_E(\boldsymbol{x}_c) + P_E^\perp(\boldsymbol{x}_c). \tag{6.17}$$

The names "projection" and "rejection" stem from the geometrical meaning of these operators. The first returns the component of $\boldsymbol{x}_c$ that is parallel to E by a projection (dot product). The latter produces the component of $\boldsymbol{x}_c$ that is orthogonal to $\boldsymbol{E}$, hence the name (see Fig. 6.1).

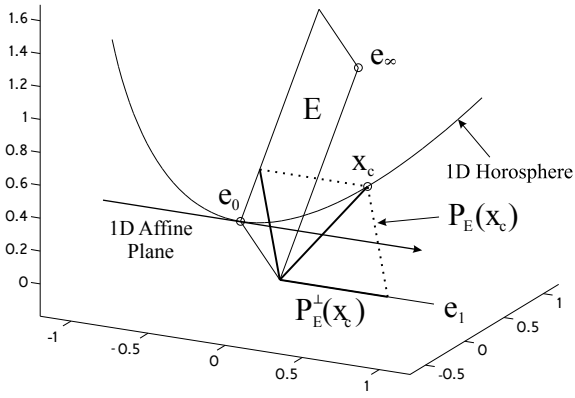


Fig. 6.1 The projection and rejection of vector $x_c \in \mathbb{R}^{2,1}$ from the $\boldsymbol{E}$ plane. The operators are illustrated for the 1D case

Hestenes introduced earlier as the additive split [1] the *multiplicative split*: $\mathbb{R}^{n,1} = \mathbb{R}^n \otimes \mathbb{R}^{1,1}$, which relates conformal and vector space models (covariant Euclidean geometry) [3, 4].

For the case of $G_{4,1}$, the null vector e_0 is assigned to the origin, so that each point lies on the bundle of all lines crossing through the origin. G_3 is the geometry of that bundle, and each point is represented by the vector $\boldsymbol{x} \in \mathbb{R}^3$. The generating basis vectors for G_3 are trivectors in $G_{4,1}$:

$$e_i = e_i \wedge e_\infty \wedge e_0 = e_i(e_\infty \wedge e_0) = e_i \boldsymbol{E} = \boldsymbol{E} e_i, \tag{6.18}$$

$$i = e_1 e_2 e_3 = (e_1 \boldsymbol{E})(e_2 \boldsymbol{E})(e_3 \boldsymbol{E}) = (e_1 e_2 e_3)\boldsymbol{E} = I_c. \tag{6.19}$$

Note that the pseudoscalar is invariant, i.e., $i = I_c$, where $i^2 = -1$. By the case of additive split, the vector basis of $e_1, e_2, e_3 \in G_3$ is not associated with lines or the origin e_0 and its pseudoscalar $I_3 = e_1 e_2 e_3$ is not invariant.

6.1.2 Conformal Splits for Points and Simplexes

We will next give the expressions using conformal splits of the basic geometric entities depicted in Fig. 6.2. The point is given by

$$\boldsymbol{x} = \left(\mathbf{x} + \frac{1}{2}\mathbf{x}^2 e_\infty + e_0\right)\boldsymbol{E} = \boldsymbol{E}\left(\mathbf{x} - \frac{1}{2}\mathbf{x}^2 e_\infty - e_0\right) = \mathbf{x}\boldsymbol{E} + \frac{1}{2}\mathbf{x}^2 e_\infty - e_0. \tag{6.20}$$

This fulfills

$$\mathbf{x} = \boldsymbol{x} \wedge e_0 \wedge e_\infty = \boldsymbol{x} \wedge \boldsymbol{E}. \tag{6.21}$$

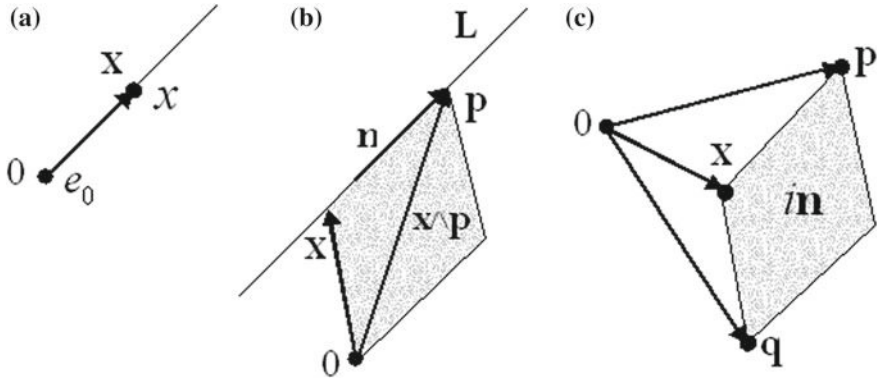


Fig. 6.2 Conformal multiplicative splits: point, line, and a plane

The line or *spear* is given by

$$\boldsymbol{L} = \boldsymbol{x} \wedge \boldsymbol{p} \wedge e_\infty = \mathbf{x} \wedge \mathbf{p} e_\infty + (\mathbf{p} - \mathbf{x}) = (\mathbf{d} e_\infty + 1)\mathbf{n}, \tag{6.22}$$

where the Plücker coordinates are given by $\mathbf{n} = (\mathbf{p} - \mathbf{x})$ (tangent), $\mathbf{x} \wedge \mathbf{p} = \mathbf{x} \wedge (\mathbf{p} - \mathbf{x}) = \mathbf{dn}$ (moment) and $\mathbf{d} = (\mathbf{x} \wedge \mathbf{p})\mathbf{n}^{-1} = \mathbf{x} - (\mathbf{x} \cdot \mathbf{n}^{-1})\mathbf{n}$ (directance). The plane is given by

$$\boldsymbol{P} = \boldsymbol{x} \wedge \boldsymbol{p} \wedge \boldsymbol{q} \wedge e_\infty = \mathbf{x} \wedge \mathbf{p} \wedge \mathbf{q} e_\infty + (\mathbf{p} - \mathbf{x}) \wedge (\mathbf{q} - \mathbf{x})\boldsymbol{E}, \tag{6.23}$$

where the tangent is $(\mathbf{p} - \mathbf{x}) \wedge (\mathbf{q} - \mathbf{x}) = \mathbf{x} \wedge \mathbf{p} + \mathbf{p} \wedge \mathbf{q} + \mathbf{q} \wedge \mathbf{x} = i\mathbf{n}$, the moment $\mathbf{x} \wedge \mathbf{p} \wedge \mathbf{q} = \mathbf{x} \wedge [(\mathbf{p} - \mathbf{x}) \wedge (\mathbf{q} - \mathbf{x})] = \mathbf{x} \wedge (i\mathbf{n}) = i(\mathbf{x} \cdot \mathbf{n})$.

The dual form of the plane is given by

$$\boldsymbol{P} = i(\mathbf{x} \cdot \mathbf{n} e_\infty + \mathbf{n}\boldsymbol{E}) = i\boldsymbol{n}, \tag{6.24}$$

where

$$\begin{aligned} \boldsymbol{n} = \boldsymbol{x}_2 - \boldsymbol{x}_1 &= (\mathbf{x}_2 - \mathbf{x}_1)\boldsymbol{E} + \frac{1}{2}(\mathbf{x}_2^2 - \mathbf{x}_1^2)e_\infty, \\ &= (\mathbf{x}_2 - \mathbf{x}_1)\boldsymbol{E} + \frac{1}{2}(\mathbf{x}_2 + \mathbf{x}_1) \cdot (\mathbf{x}_2 - \mathbf{x}_1)e_\infty = \mathbf{n}\boldsymbol{E} + \mathbf{c} \cdot \mathbf{n} e_\infty. \end{aligned} \tag{6.25}$$

In this book, we have use more the additive split than the multiplicative split; however, the use of the multiplicative split and its representations are matter of our future works.

6.1.3 Euclidean and Conformal Spaces

One of the results of the non-Euclidean geometry demonstrated by Nikolai Lobachevsky in the nineteenth century is that in spaces with hyperbolic structure we can find subsets that are isomorphic to a Euclidean space. In order to do this, Lobachevsky introduced two constraints, to what we now call *conformal point* $\boldsymbol{x}_c \in \mathbb{R}^{n+1,1}$, see Fig. 6.3.

The first constraint is the *homogeneous* representation, *normalizing* the vector $\boldsymbol{x}_c$ such that

$$\boldsymbol{x}_c \cdot e_\infty = -1, \tag{6.26}$$

and the second constraint is that the vector must be a *null vector*, that is,

$$\boldsymbol{x}_c^2 = 0. \tag{6.27}$$

Now, recall that a hyperplane $\mathbb{P}(\boldsymbol{n}, \boldsymbol{a}) \in \mathbb{R}^{n+1,1}$ with normal $\boldsymbol{n}$ and passing through the point $\boldsymbol{a}$ is the solution to the equation

$$\boldsymbol{n} \cdot (\boldsymbol{x} - \boldsymbol{a}) = 0, \ \boldsymbol{x} \in \mathbb{R}^{n+1,1}. \tag{6.28}$$

The normalization condition $\boldsymbol{x}_c \cdot e_\infty = e_\infty \cdot e_0 = -1$ is equivalent to the equation

$$e_\infty \cdot (\boldsymbol{x}_c - e_0) = 0, \tag{6.29}$$

which is the equation of a hyperplane $\mathbb{P}(e_\infty, e_0)$. Thus, the normalization condition of Eq. (6.26) constraints the points $\boldsymbol{x}_c$ to lie in a hyperplane passing though e_0 with

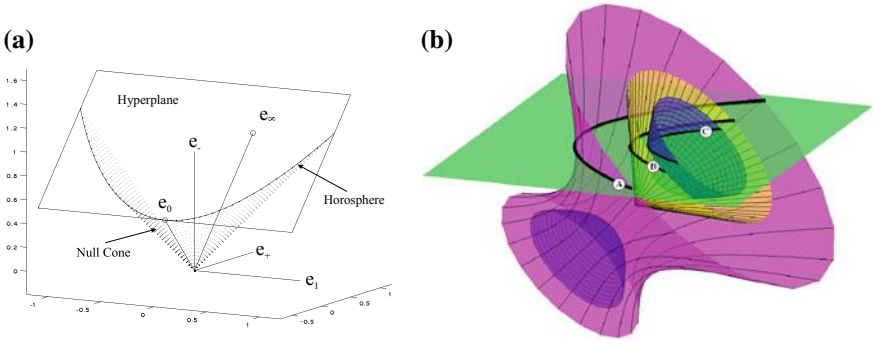


Fig. 6.3 **a** The null cone (dotted lines), the hyperplane $\mathbb{P} + (e_\infty, e_0)$, and the horosphere for 1-D. Note that even though the normal of the hyperplane is e_∞, the plane is actually *geometrically parallel* to this vector. **b** Surface levels A, B, and C denoting spheres with a radius positive, zero (null cone), and negative, radius, respectively

normal e_∞. Equation (6.26) fixes the scale; however, for the conformal model, another constraint is needed to fix $\boldsymbol{x}_c$ as a unique representation of $\mathbf{x}_e \in \mathbb{R}^n$.

Note that the inner product of two conformal points

$$\begin{aligned} \boldsymbol{x} \cdot \boldsymbol{y} &= \left(\mathbf{x} + \frac{1}{2}\mathbf{x}^2 e_\infty + e_0\right) \cdot \left(\mathbf{y} + \frac{1}{2}\mathbf{y}^2 e_\infty + e_0\right), \\ &= \mathbf{x} \cdot \mathbf{y} - \frac{1}{2}\mathbf{x}^2 - \frac{1}{2}\mathbf{y}^2 = -\frac{1}{2}(\mathbf{x} - \mathbf{y})^2, \\ &= -\frac{1}{2}\|\mathbf{x} - \mathbf{y}\|^2 \end{aligned} \tag{6.30}$$

yields a quadratic representation of the Euclidean distance between the two Euclidean points: $\mathbf{x}$ and $\mathbf{y}$.

To complete the definition of *generalized homogeneous coordinates* for points in $\mathbb{R}^{n+1,1}$, we resort therefore to the second constraint of Eq. (6.27) $\boldsymbol{x}_c^2 = 0$. The set $\mathbb{N}^{n+1}$ of vectors that square to zero is called the *null cone*. Therefore, conformal points are required to lie in the intersection of the null cone $\mathbb{N}^{n+1}$ with the hyperplane $\mathbb{P}(e_\infty, e_0)$. The resulting surface $\mathbb{N}_e^n$ is called the *horosphere*:

$$\mathbb{N}_e^n = \mathbb{N}^{n+1} \cap \mathbb{P}(e_\infty, e_0) = \{\boldsymbol{x}_c \in \mathbb{R}^{n+1,1} | \boldsymbol{x}_c^2 = 0, \boldsymbol{x}_c \cdot e_\infty = -1\}. \tag{6.31}$$

The homogeneous model horosphere has its origins in the work of F. A.: Wachter (1792–1817) a student of Gauss [5].

An illustration of the null cone, the hyperplane, and the horosphere can be seen in Fig. 6.3a. By values of $\boldsymbol{x}_c^2$ that are positive, zero (null cone), and negative, three families of surfaces are obtained as shown in Fig. 6.3b.

The constraints (6.27) and (6.26) now included in (6.31) define an isomorphic mapping between the Euclidean space and the conformal space. Thus, for each conformal point $\boldsymbol{x}_c \in \mathbb{R}^{n+1,1}$ there are a unique Euclidean space point $\boldsymbol{x}_e \in \mathbb{R}^n$ and unique scalars α, β such that the following mapping is bijective:

$$\boldsymbol{x}_e \mapsto \boldsymbol{x}_c = \boldsymbol{x}_e + \alpha e_0 + \beta e_\infty. \tag{6.32}$$

To see how this mapping is obtained, first we see that any point $\boldsymbol{x}_c = \mathbf{x}_e + \alpha e_0 + \beta e_\infty \in \mathbb{N}_e^n$ can be expressed as $\boldsymbol{x}_c = \mathbf{x}_e + k_1 e_+ + k_2 e_-$, for some scalars k_1, k_2, since e_0 and e_∞ are linear combinations of the basis vectors e_+ and e_-. Since $\boldsymbol{E}^2 = 1$, we can apply the conformal split to x_c, and we get

$$\boldsymbol{x}_c = \boldsymbol{x}_c \boldsymbol{E}^2 = (\boldsymbol{x}_c \wedge \boldsymbol{E} + \boldsymbol{x}_c \cdot \boldsymbol{E})\boldsymbol{E} = (\boldsymbol{x}_c \wedge \boldsymbol{E})\boldsymbol{E} + (\boldsymbol{x}_c \cdot \boldsymbol{E})\boldsymbol{E}, \tag{6.33}$$

see Fig. 6.3. Now, recall that $(\boldsymbol{x}_c \wedge \boldsymbol{E})\boldsymbol{E} = \mathbf{x}_e$ is the rejection (see Eq. (6.17)). The expression $(x_c \cdot \boldsymbol{E})\boldsymbol{E}$ can be expanded as

$$(\boldsymbol{x}_c \cdot \boldsymbol{E})\boldsymbol{E} = (\boldsymbol{x}_c \cdot (e_\infty \wedge e_0))\boldsymbol{E} = e_0 + (k_1 + k_2)e_\infty. \tag{6.34}$$

Now, applying the condition that $\boldsymbol{x}_c^2 = 0$, we find from Eq. (6.33) that

$$\begin{aligned} \boldsymbol{x}_c^2 &= ((\boldsymbol{x}_c \wedge \boldsymbol{E})\boldsymbol{E} + (\boldsymbol{x}_c \cdot \boldsymbol{E})\boldsymbol{E})^2, \\ 0 &= (\mathbf{x}_e + e_0 + (k_1 + k_2)e_\infty)^2 = \mathbf{x}_e^2 - (k_1 + k_2), \\ \mathbf{x}_e^2 &= (k_1 + k_2). \end{aligned} \tag{6.35}$$

Finally, using Eq. (6.33), and substituting Eq. (6.35) into Eq. (6.34), we get

$$\boldsymbol{x}_c = (\boldsymbol{x}_c \wedge \boldsymbol{E})\boldsymbol{E} + (\boldsymbol{x}_c \cdot \boldsymbol{E})\boldsymbol{E} = \mathbf{x}_e + e_0 + \frac{1}{2}(k_1 + k_2)e_\infty = \mathbf{x}_e + \frac{1}{2}\mathbf{x}_e^2 e_\infty + e_0. \tag{6.36}$$

We can gain further insight into the geometrical meaning of the null vectors by analyzing Eq. (6.36). For instance, by setting $\mathbf{x}_e = 0$, we find that e_0 represents the origin of $\mathbb{R}^n$ (hence the name). Similarly, dividing this equation by $\boldsymbol{x}_c \cdot e_0 = -\frac{1}{2}\mathbf{x}_e^2$ gives

$$\begin{aligned} \frac{\boldsymbol{x}_c}{\boldsymbol{x}_c \cdot e_0} &= -\frac{2}{\mathbf{x}_e^2}\left(\mathbf{x}_e + \frac{1}{2}\mathbf{x}_e^2 e_\infty + e_0\right) = -\frac{2\mathbf{x}_e^2}{\mathbf{x}_e^2}\left(\frac{1}{\mathbf{x}_e} + \frac{1}{2}e_\infty + \frac{e_0}{\mathbf{x}_e^2}\right), \\ &= -2\left(\frac{1}{\mathbf{x}_e} + \frac{1}{2}e_\infty + \frac{e_0}{\mathbf{x}_e^2}\right) \underset{\mathbf{x}_e \to \infty}{\longrightarrow} e_\infty. \end{aligned} \tag{6.37}$$

Thus, we conclude that e_∞ represents the point at infinity.

6.1.4 Stereographic Projection

Conformal geometry is equivalent to stereographic projection in Euclidean space. Generally speaking, a stereographic projection is a mapping, taking points lying on a hypersphere to points lying on a hyperplane and following a simple geometric construction. It is well known that this projection is used in cartography to make maps of the earth, see Fig. 6.4. In that case, the projection plane passes through the equator and the sphere is centered at the origin. To make a projection, a line is drawn from the north pole to each point on the sphere, and the intersection of this line with the projection plane constitutes the stereographic projection.

Next, we will illustrate the equivalence between stereographic projection and conformal geometric algebra in $\mathbb{R}^1$. We will be working in $\mathbb{R}^{2,1}$, with the basis vectors $\{e_1, e_+, e_-\}$ having the usual properties. The projection plane will be the x-axis, and the sphere will be a circle centered at the origin with unitary radius.

Given a scalar $\mathbf{x}_e$ representing a point on the x-axis, we wish to find the point $\boldsymbol{x}_c$ lying on the circle that projects to it (see Fig. 6.4b). The equation of the line passing through the north pole and $\mathbf{x}_e$ is given by $f(\mathbf{x}) = -\frac{1}{\mathbf{x}_e}\mathbf{x} + 1$. The equation of the circle is $\mathbf{x}^2 + f(\mathbf{x})^2 = 1$. Substituting the equation of the line on the circle, we get $\mathbf{x}^2 - 2\mathbf{x}\mathbf{x}_e + \mathbf{x}^2\mathbf{x}_e^2 = 0$, which has the two solutions $\mathbf{x} = 0, \quad \mathbf{x} = 2\frac{\mathbf{x}_e}{\mathbf{x}_e^2+1}$. Only the

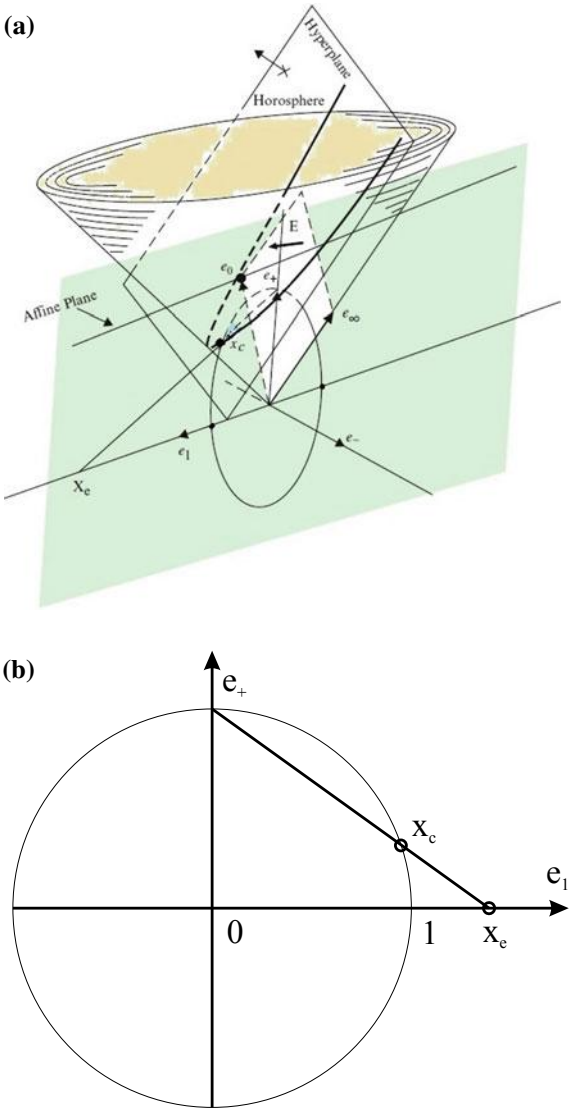


Fig. 6.4 **a** Null cone, hyperplane, horosphere, and affine plane for the 1D case. **b** Stereographic projection for the 1D case

latter solution is meaningful. Substituting in the equation of the line, we get $f(\mathbf{x}) = \frac{\mathbf{x}_e^2-1}{\mathbf{x}_e^2+1}$. Hence, $\boldsymbol{x}_c$ has coordinates $\boldsymbol{x}_c = \left(2\frac{\mathbf{x}_e}{\mathbf{x}_e^2+1}, \frac{\mathbf{x}_e^2-1}{\mathbf{x}_e^2+1}\right)$, which can be represented in homogeneous coordinates as the vector

$$\boldsymbol{x}_c = 2\frac{\mathbf{x}_e}{\mathbf{x}_e^2+1}e_1 + \frac{\mathbf{x}_e^2-1}{\mathbf{x}_e^2+1}e_+ + e_-. \tag{6.38}$$

If we take the limits to Eq. (6.38), we get the expected points at infinity and the origin of $\mathbb{R}^n$:

$$\lim_{\mathbf{x}_e \to \infty} \boldsymbol{x}_c = e_+ + e_- = e_\infty, \tag{6.39}$$
$$\lim_{\mathbf{x}_e \to 0} \frac{\boldsymbol{x}_c}{2} = \frac{e_- - e_+}{2} = e_0.$$

This result is a first confirmation that stereographic projection is equivalent to a conformal mapping given by Eq. (6.36). For a second proof, we note that Eq. (6.38) can be rewritten as

$$\begin{aligned} \boldsymbol{x}_c &= 2\frac{\mathbf{x}_e}{\mathbf{x}_e^2 + 1}e_1 + \frac{\mathbf{x}_e^2 - 1}{\mathbf{x}_e^2 + 1}e_+ + e_-, \\ &= \frac{2}{\mathbf{x}_e^2 + 1}\left(\mathbf{x}_e e_1 + \frac{1}{2}(\mathbf{x}_e^2 - 1)e_+ + \frac{1}{2}(\mathbf{x}_e^2 + 1)e_-\right). \end{aligned} \tag{6.40}$$

Dividing by the scale factor $\frac{2}{\mathbf{x}_e^2+1}$ in order to achieve the constraint imposed by Eq. (6.26)—$\boldsymbol{x}_c \cdot e_\infty = -1$, we arrive at

$$\begin{aligned} \boldsymbol{x}_c &= \mathbf{x}_e e_1 + (\mathbf{x}_e^2 - 1)e_+ + (\mathbf{x}_e^2 + 1)e_- = \mathbf{x}_e e_1 + \frac{1}{2}\mathbf{x}_e^2 e_\infty + e_0, \\ &= \mathbf{x}_e + \frac{1}{2}\mathbf{x}_e^2 e_\infty + e_0, \end{aligned} \tag{6.41}$$

where $\mathbf{x}_e = x_e e_1$, which is precisely Eq. (6.36). Hence, we have demonstrated that conformal geometric algebra is projectively equivalent to a stereographic projection (i.e., *up to a scale factor*).

6.1.5 *Inner and Outer Product Null Spaces*

For the next definitions, we adopt the classification of geometric entities introduced by Ch. Perwass [6]. We will consider that given a vector $\boldsymbol{x}$ if its inner or wedge product with a blade $\boldsymbol{X}_k$ is zero, we can consider that $\boldsymbol{x}$ with respect to these operations belongs to the null space of the blade $\boldsymbol{X}_k$, see a more details of this definition in [6, 7].

Thus, the *Inner Product Null Space* (IPNS) of a blade $\boldsymbol{X}_k \in G^k_{p,q}$, denoted by $\mathbb{NI}(\boldsymbol{X}_k)$, is defined as

$$\mathbb{NI}(\boldsymbol{X}_k) := \{\boldsymbol{x} \in G^1_{p,q} : \boldsymbol{x} \cdot \boldsymbol{X}_k = 0\}, \tag{6.42}$$

regardless if the blade is a null space or not.

Thinking in the dual space, we can define the *Outer Product Null Space* (OPNS) of a blade $\boldsymbol{X}_k \in G^k_{p,q}$, denoted by $\mathbb{NO}(\boldsymbol{X}_k)$, as follows:

$$\mathbb{NO}(\boldsymbol{X}_k) := \{\boldsymbol{x} \in G^1_{p,q} : \boldsymbol{x} \wedge \boldsymbol{X}_k = 0\}, \tag{6.43}$$

regardless if the blade is a null space or not.

The dual operation on blades can be best seen considering the dual relation between the IPNS and OPNS of the blade. Let $\boldsymbol{x} \in G^1_{p,q}$, $\boldsymbol{X}_k \in G^k_{p,q}$ with $k \geq 1$ and the pseudoscalar I of $G_{p,q}$; according to equations given in Sect. 2.4.2, we can formulate the dual relationship as follows:

$$(\boldsymbol{x} \wedge \boldsymbol{X}_k)^* = (\boldsymbol{x} \wedge \boldsymbol{X}_k) \cdot I^{-1} = \boldsymbol{x} \cdot \boldsymbol{X}^*_k, \tag{6.44}$$

hence,

$$\boldsymbol{x} \wedge \boldsymbol{X}_k = 0 \iff \boldsymbol{x} \cdot \boldsymbol{X}^*_k = 0, \tag{6.45}$$

and thus

$$\mathbb{NO}(\boldsymbol{X}_k) = \mathbb{NI}(\boldsymbol{X}_k). \tag{6.46}$$

6.1.6 Spheres and Planes

The equation of a sphere of radius ρ centered at point $\mathbf{c}_e \in \mathbb{R}^n$ can be written as

$$(\mathbf{x}_e - \mathbf{c}_e)^2 = \rho^2. \tag{6.47}$$

Since $\boldsymbol{x}_c \cdot \boldsymbol{y}_c = -\frac{1}{2}(\mathbf{x}_e - \mathbf{y}_e)^2$, we can rewrite the above formula in terms of homogeneous coordinates as

$$\boldsymbol{x}_c \cdot \boldsymbol{c}_c = -\frac{1}{2}\rho^2. \tag{6.48}$$

Since $\boldsymbol{x}_c \cdot e_\infty = -1$, we can factor the above expression and then

$$\boldsymbol{x}_c \cdot \left(\boldsymbol{c}_c - \frac{1}{2}\rho^2 e_\infty\right) = 0. \tag{6.49}$$

This equation corresponds to the IPNS representation and yields finally the simplified equation for the sphere as

$$\boldsymbol{x}_c \cdot \boldsymbol{s} = 0, \tag{6.50}$$

where

$$\boldsymbol{s} = \boldsymbol{c}_c - \frac{1}{2}\rho^2 e_\infty = \mathbf{c}_e + e_0 + \frac{\mathbf{c}_e^2 - \rho^2}{2} e_\infty \tag{6.51}$$

is the equation of the sphere. From this equation and (6.36), we can see that a conformal point is just a sphere with zero radius. The vector s has the properties

$$\boldsymbol{s}^2 = \rho^2 > 0, \tag{6.52}$$

$$e_\infty \cdot \boldsymbol{s} = -1. \tag{6.53}$$

From these properties, we conclude that the sphere $\boldsymbol{s}$ is a point lying on the hyperplane $\boldsymbol{x}_c \cdot e_\infty = -1$, but *outside* the null cone $\boldsymbol{x}_c^2 = 0$. In particular, all points on the hyperplane outside the horosphere determine spheres with positive radius, points lying on the horosphere define spheres of zero radius (i.e., points), and points lying inside the horosphere have imaginary radius. Finally, note that spheres of the same radius form a surface that is parallel to the horosphere.

Alternatively, spheres can be dualized and represented as $(n+1)$-vectors $\boldsymbol{s}^* = \boldsymbol{s}I^{-1}$. Then using the *main conjugation* $\widetilde{I}$ of I defined as

$$\widetilde{I} = (-1)^{\frac{1}{2}(n+2)(n+1)} I = -I^{-1}. \tag{6.54}$$

We can express the constraints of Eqs. (6.52) and (6.53) as

$$\begin{aligned} \boldsymbol{s}^2 &= -\widetilde{\boldsymbol{s}^*}\boldsymbol{s}^* = \rho^2, \\ e_\infty \cdot \boldsymbol{s} &= e_\infty \cdot (\boldsymbol{s}^* I) = (e_\infty \wedge \boldsymbol{s}^*) I = -1. \end{aligned} \tag{6.55}$$

Similar as in Eq. (6.50), the equation involving a dual sphere reads

$$\boldsymbol{x}_c \wedge \boldsymbol{s}^* = 0. \tag{6.56}$$

The advantage of the dual form is that the sphere can be directly computed from four points (in 3D) yielding the OPNS representation for the sphere

$$\boldsymbol{s}^* = \boldsymbol{x}_{c_1} \wedge \boldsymbol{x}_{c_2} \wedge \boldsymbol{x}_{c_3} \wedge \boldsymbol{x}_{c_4}. \tag{6.57}$$

If we replace one of these points for the point at infinity, we get

$$\pi^* = \boldsymbol{x}_{c_1} \wedge \boldsymbol{x}_{c_2} \wedge \boldsymbol{x}_{c_3} \wedge e_\infty. \tag{6.58}$$

In standard IPNS form, π is given by

$$\pi = I\pi^* = n + de_\infty, \tag{6.59}$$

where n is the normal vector and d represents the Hesse distance for the 3D space.

Developing the products, we get the OPNS representation for the plane

$$\begin{aligned} \pi^* &= \boldsymbol{x}_{c_3} \wedge \boldsymbol{x}_{c_1} \wedge \boldsymbol{x}_{c_2} \wedge e_\infty, \\ &= \mathbf{x}_{e_3} \wedge \mathbf{x}_{e_1} \wedge \mathbf{x}_{e_2} \wedge e_\infty + ((\mathbf{x}_{e_3} - \mathbf{x}_{e_1}) \wedge (\mathbf{x}_{e_2} - \mathbf{x}_{e_1}))\boldsymbol{E}, \end{aligned} \tag{6.60}$$

which is the equation of the plane passing through the points $\mathbf{x}_{e_1}$, $\mathbf{x}_{e_2}$, and $\mathbf{x}_{e_3}$. We can easily see that $\mathbf{x}_{e_1} \wedge \mathbf{x}_{e_2} \wedge \mathbf{x}_{e_3}$ is a pseudoscalar representing the volume of the parallelepiped with sides $\mathbf{x}_{e_1}$, $\mathbf{x}_{e_2}$ and $\mathbf{x}_{e_3}$. Also, since $(\mathbf{x}_{e_1} - \mathbf{x}_{e_2})$ and $(\mathbf{x}_{e_3} - \mathbf{x}_{e_2})$ are two vectors on the plane, the expression $((\mathbf{x}_{e_1} - \mathbf{x}_{e_2}) \wedge (\mathbf{x}_{e_3} - \mathbf{x}_{e_2}))^*$ is the normal to the plane. Therefore, planes are spheres passing through the point at infinity.

6.1.7 Geometric Identities, Dual, Meet, and Join Operations

A circle z can be regarded as the intersection of two spheres $\boldsymbol{s}_1$ and $\boldsymbol{s}_2$. This means that for each point on the circle $\boldsymbol{x}_c \in z$ they lie on both spheres, that is, $\boldsymbol{x}_c \in \boldsymbol{s}_1$ and $\boldsymbol{x}_c \in \boldsymbol{s}_2$. Assuming that $\boldsymbol{s}_1$ and $\boldsymbol{s}_2$ are linearly independent, we can write for $\boldsymbol{x}_c \in z$

$$(\boldsymbol{x}_c \cdot \boldsymbol{s}_1)\boldsymbol{s}_2 - (\boldsymbol{x}_c \cdot \boldsymbol{s}_2)\boldsymbol{s}_1 = x_c \cdot (\boldsymbol{s}_1 \wedge \boldsymbol{s}_2) = \boldsymbol{x}_c \cdot z = 0. \tag{6.61}$$

This result tells us that since $\boldsymbol{x}_c$ lies on both spheres, $z = (\boldsymbol{s}_1 \wedge \boldsymbol{s}_2)$ should be the intersection of the spheres or a circle. It is easy to see that the intersection with a third sphere leads to a point pair. We have derived algebraically that the wedge of two linearly independent spheres yields to their intersecting circle (see Fig. 6.5b). This topological relationship between two spheres can be also conveniently described using the dual of the meet operation, namely,

$$z = (z^*)^* = (\boldsymbol{s}_1^* \vee \boldsymbol{s}_2^*)^* = \boldsymbol{s}_1 \wedge \boldsymbol{s}_2. \tag{6.62}$$

This new equation says that the dual of a circle can be computed via the meet of two spheres in their dual form. This equation confirms geometrically our previous algebraic computation of Eq. (6.61).

The standard OPNS (dual) form of the circle (in 3D) can be expressed by three points lying on it as

$$z^* = \boldsymbol{x}_{c_1} \wedge \boldsymbol{x}_{c_2} \wedge \boldsymbol{x}_{c_3}, \tag{6.63}$$

see Fig. 6.5a. Similar to the case of planes shown in Eq. (6.58), lines can be defined by circles passing through the point at infinity as

$$\boldsymbol{l}^* = \boldsymbol{x}_{c_1} \wedge \boldsymbol{x}_{c_2} \wedge e_\infty. \tag{6.64}$$

This can be demonstrated by developing the wedge products as in the case of the planes to yield the OPNS representation of the line:

$$\boldsymbol{x}_{c_1} \wedge \boldsymbol{x}_{c_2} \wedge e_\infty = \mathbf{x}_{e_1} \wedge \mathbf{x}_{e_2} \wedge e_\infty + (\mathbf{x}_{e_2} - \mathbf{x}_{e_1}) \wedge \boldsymbol{E}, \tag{6.65}$$

from where it is evident that the expression $\mathbf{x}_{e_1} \wedge \mathbf{x}_{e_2}$ is a bivector representing the plane where the line is contained and $(\mathbf{x}_{e_2} - \mathbf{x}_{e_1})$ is the direction of the line.

The standard IPNS form of the line can be expressed as

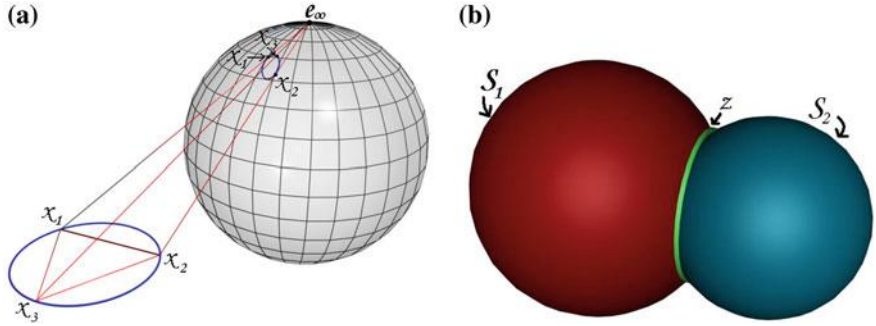


Fig. 6.5 **a** (*left*) Circle computed using three points, note its stereographic projection. **b** (*right*) Circle computed using the meet of two spheres

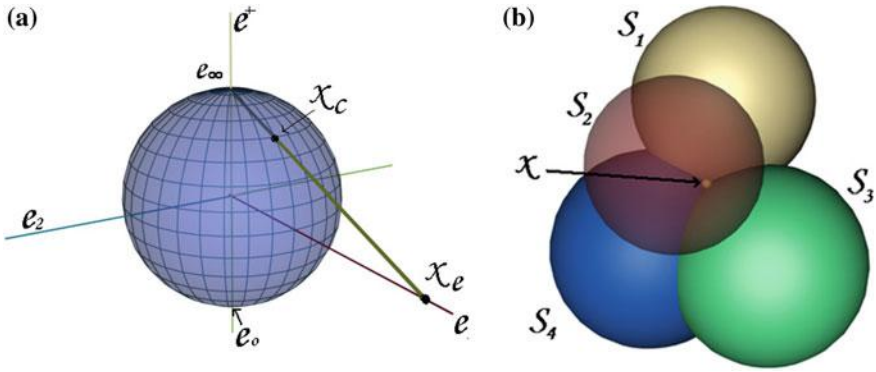


Fig. 6.6 **a** (*left*) Conformal point generated by projecting a point of the affine plane to the unit sphere. Note that we use only the Riemann sphere and the 2D plane. **b** (*right*) Point generated by the meet of four spheres

$$L = \boldsymbol{n} I_e - e_\infty \boldsymbol{m} I_e, \tag{6.66}$$

where $\boldsymbol{n}$ and $\boldsymbol{m}$ stand for the line orientation and moment, respectively. The line in the IPNS standard form is a bivector representing the six Plücker coordinates.

The dual of a point $\boldsymbol{x}$ is a sphere $\boldsymbol{s}$. The intersection of four spheres yields the OPNS equation for the point: see Fig. 6.6b. The dual relationships between a point and its dual, the sphere, are

$$\boldsymbol{s}^* = \boldsymbol{x}_{c_1} \wedge \boldsymbol{x}_{c_2} \wedge \boldsymbol{x}_{c_3} \wedge \boldsymbol{x}_{c_4} \leftrightarrow \boldsymbol{x}^* = \boldsymbol{s}_1 \wedge \boldsymbol{s}_2 \wedge \boldsymbol{s}_3 \wedge \boldsymbol{s}_4, \tag{6.67}$$

where the points are denoted as $\boldsymbol{x}_i$ and the spheres $\boldsymbol{s}_i$ for $i = 1, 2, 3, 4$.

There is another very useful relationship between an $(r-2)$-dimensional sphere $\boldsymbol{A}_r$ and the sphere $\boldsymbol{s}^*$ (computed as the dual of a point $\boldsymbol{s}$). If from the sphere $\boldsymbol{A}_r$ we can compute the hyperplane $\boldsymbol{A}_{r+1} \equiv \boldsymbol{e}_\infty \wedge \boldsymbol{A}_r \neq 0$, we can express the meet between the

dual of the point $\boldsymbol{s}$ (a sphere) and the hyperplane $\boldsymbol{A}_{r+1}$ as getting the sphere $\boldsymbol{A}_r$ of one dimension lower:

$$(-1)^{\epsilon}\boldsymbol{s}^* \cap \boldsymbol{A}_{r+1} = (\boldsymbol{s}^* I) \cdot \boldsymbol{A}_{r+1} = \boldsymbol{s}\boldsymbol{A}_{r+1} = \boldsymbol{A}_r. \tag{6.68}$$

This result is telling us an interesting relationship: that the sphere $\boldsymbol{A}_r$ and the hyperplane $\boldsymbol{A}_{r+1}$ are related via the point $\boldsymbol{s}$ (dual of the sphere $\boldsymbol{s}^*$); thus, we then rewrite Eq. (6.68) as follows:

$$\boldsymbol{s} = \boldsymbol{A}_r \boldsymbol{A}_{r+1}^{-1}. \tag{6.69}$$

Using Eq. (6.69) and given the plane π (A_{r+1}) and the circle $\boldsymbol{z}$ ($\boldsymbol{A}_r$), we can compute the sphere

$$\boldsymbol{s} = \boldsymbol{z}\pi^{-1}. \tag{6.70}$$

Similarly, we can compute another important geometric relationship called the *pair of points* using the Eq. (6.69) directly:

$$\boldsymbol{s} = \boldsymbol{P}\boldsymbol{P}\boldsymbol{L}^{-1}. \tag{6.71}$$

Now, using this result given the line $\boldsymbol{L}$ and the sphere $\boldsymbol{s}$, we can compute the IPNS form of the pair of points $\boldsymbol{P}\boldsymbol{P}$ (see Fig. 6.7b):

$$\boldsymbol{P}\boldsymbol{P} = \boldsymbol{s}\boldsymbol{L} = \boldsymbol{s} \wedge \boldsymbol{L}. \tag{6.72}$$

The OPNS of the pair the points is given by

$$\boldsymbol{P}\boldsymbol{P}^* = \boldsymbol{x}_{c_1} \wedge \boldsymbol{x}_{c_2}. \tag{6.73}$$

A summary of the standard IPNS and OPNS forms of the basic geometric entities is presented in Table 6.1.

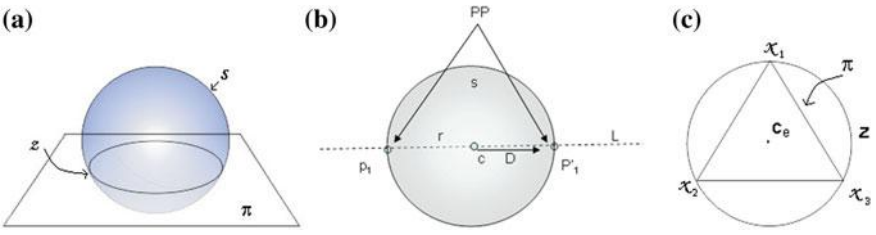


Fig. 6.7 **a** The meet of a sphere and a plane. **b** Pair of points resulting from the meet between a line and a sphere. **c** Center of a circumscribed triangle

Table 6.1 Representation of entities in conformal geometric algebra, G = grade

Entity	IPNS	G	OPNS (dual)	G
Sphere	$\boldsymbol{s} = \mathbf{c} + \frac{1}{2}(\mathbf{c}^2 - \rho^2)e_\infty + e_0$	1	$\boldsymbol{s}^* = \boldsymbol{x}_{c_1} \wedge \boldsymbol{x}_{c_2} \wedge \boldsymbol{x}_{c_3} \wedge \boldsymbol{x}_{c_4}$	4
Point	$\boldsymbol{x}_c = \mathbf{x} + \frac{1}{2}\mathbf{x}^2 e_\infty + e_0$	1	$\boldsymbol{x}^* = \boldsymbol{s}_1 \wedge \boldsymbol{s}_2 \wedge \boldsymbol{s}_3 \wedge \boldsymbol{s}_4$	4
Plane	$\pi = \boldsymbol{n} I_E - d e_\infty$	1	$\pi^* = e_\infty \wedge \boldsymbol{x}_{c_1} \wedge \boldsymbol{x}_{c_2} \wedge \boldsymbol{x}_{c_3}$	4
	$\boldsymbol{n} = (\mathbf{x}_{e_1} - \mathbf{x}_{e_2}) \wedge (\mathbf{x}_{e_1} - \mathbf{x}_{e_3})$			
	$d = (\mathbf{x}_{e_1} \wedge \mathbf{x}_{e_2} \wedge \mathbf{x}_{e_3}) I_E$			
Line	$\boldsymbol{L} = \pi_1 \wedge \pi_2$	2	$\boldsymbol{L}^* = e_\infty \wedge \boldsymbol{x}_{c_1} \wedge \boldsymbol{x}_{c_2}$	3
	$\boldsymbol{L} = \boldsymbol{n} I_E - e_\infty \boldsymbol{m} I_E$			
	$\boldsymbol{n} = (\mathbf{x}_{e_1} - \mathbf{x}_{e_2})$			
	$\boldsymbol{m} = (\mathbf{x}_{e_1} \wedge \mathbf{x}_{e_2})$			
Circle	$z = \boldsymbol{s}_1 \wedge \boldsymbol{s}_2 = \boldsymbol{s}_1 \wedge \pi_2$	2	$\boldsymbol{z}^* = \boldsymbol{x}_{c_1} \wedge \boldsymbol{x}_{c_2} \wedge \boldsymbol{x}_{c_3}$	3
Point pair	$\boldsymbol{PP} = \boldsymbol{s}_1 \wedge \boldsymbol{s}_2 \wedge \boldsymbol{s}_3$	3	$\boldsymbol{PP}^* = \boldsymbol{x}_{c_1} \wedge \boldsymbol{x}_{c_2}$	2
	$\boldsymbol{PP} = \boldsymbol{s} \wedge \boldsymbol{L}$	2		

Now consider the following element in the conformal model:

$$\boldsymbol{X} = \alpha(e_0 \wedge \boldsymbol{x}_1 \wedge \cdots \wedge \boldsymbol{x}_k \wedge e_\infty). \tag{6.74}$$

Since $\boldsymbol{x}_i \wedge \boldsymbol{x}_j = \boldsymbol{x}_i \wedge (\boldsymbol{x}_j - \boldsymbol{x}_i)$, the previous equation can be rewritten as follows:

$$\boldsymbol{X} = \alpha(e_0 \wedge (\boldsymbol{x}_1 - e_0) \wedge \cdots \wedge (\boldsymbol{x}_k - e_0) \wedge e_\infty). \tag{6.75}$$

Now, substituting the conformal representation of the Euclidean points

$$\boldsymbol{x}_i = \mathbf{x}_i + \frac{1}{2}\mathbf{x}_i^2 e_\infty + e_0, \tag{6.76}$$

we obtain

$$\boldsymbol{X} = \alpha\left(e_0 \wedge \left(\mathbf{x}_1 + \frac{1}{2}\mathbf{x}_1^2 e_\infty\right) \wedge \cdots \wedge \left(\mathbf{x}_k + \frac{1}{2}\mathbf{x}_k^2 e_\infty\right) \wedge e_\infty\right). \tag{6.77}$$

Since the wedge product eliminates the extra terms $\frac{1}{2}\mathbf{x}_i^2 e_\infty$, we get

$$\boldsymbol{X} = \alpha(e_0 \wedge \mathbf{x}_1 \wedge \cdots \wedge \mathbf{x}_k \wedge e_\infty). \tag{6.78}$$

Thus, the part involving the vectors $\mathbf{x}_i$ and the weight α can be seen as a purely Euclidean k-blade $\mathbf{X}_k = \alpha\mathbf{x}_1 \wedge \cdots \wedge \mathbf{x}_k$; therefore, this class of blade in the conformal model is equivalent to

$$\boldsymbol{X} = e_0 \wedge \mathbf{X}_k \wedge e_\infty. \tag{6.79}$$

Finally, we can claim that the general form of a flat k-dimensional Euclidean subspace through the point $\boldsymbol{p}$ is given by

$$\boldsymbol{p} \wedge \mathbf{X}_k \wedge e_\infty. \tag{6.80}$$

6.1.8 Simplexes and Spheres

In CGA, geometric objects can be computed as the wedge of linearly independent homogeneous points $\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_r$, with $r \leq n$ so that $\boldsymbol{a}_1 \wedge \boldsymbol{a}_2 \wedge \cdots \wedge \boldsymbol{a}_r \neq 0$. This multivector can be expressed in an expanded form as follows:

$$\boldsymbol{a}_1 \wedge \boldsymbol{a}_2 \wedge \cdots \wedge \boldsymbol{a}_r = \mathbf{A}_r + e_0 \mathbf{A}_r^+ + \frac{1}{2} e_\infty \mathbf{A}_r^- - \frac{1}{2} \boldsymbol{E} \mathbf{A}_r^\pm, \tag{6.81}$$

where

$$\begin{aligned}
\mathbf{A}_r &= \mathbf{a}_0 \wedge \mathbf{a}_1 \wedge \cdots \wedge \mathbf{a}_r, \\
\mathbf{A}_r^+ &= \sum_{i=0}^{r} (-1)^i \mathbf{a}_0 \wedge \cdots \wedge \check{\mathbf{a}}_i \wedge j \cdots \wedge \mathbf{a}_r = \big(\mathbf{a}_1 - \mathbf{a}_0\big) \wedge \cdots \wedge \big(\mathbf{a}_r - \mathbf{a}_0\big), \\
\mathbf{A}_r^- &= \sum_{i=0}^{r} (-1)^i \mathbf{a}_i^2 \mathbf{a}_0 \wedge \cdots \wedge \check{\mathbf{a}}_i \wedge \cdots \wedge \mathbf{a}_r, \\
\mathbf{A}_r^\pm &= \sum_{i=0}^{r} \sum_{j=i+1}^{r} (-1)^{i+j} \big(\mathbf{a}_i^2 - \mathbf{a}_j^2\big) \mathbf{a}_0 \wedge \cdots \wedge \check{\mathbf{a}}_i \wedge \cdots \wedge \check{\mathbf{a}}_j \wedge \cdots \wedge \mathbf{a}_r.
\end{aligned} \tag{6.82}$$

The expanded form of Eq. (6.81) gives the following geometric information:

- determines an r-simplex if $\mathbf{A}_r \neq 0$,
- represents an $(r-1)$-simplex in a plane which passes through the origin if $\mathbf{A}_r^+ = \mathbf{A}_r^- = 0$, and
- represents an $(r-1)$-sphere if and only if $\mathbf{A}_r^+ \neq 0$.

See [1] for a more detailed study on the expanded form.

In Eq. (6.81), $\mathbf{A}_r$ is the moment of the simplex with a *boundary* (or *tangent*) $\mathbf{A}_r^+$, and thus the corresponding r-simplex can be formulated as follows:

$$e \wedge a_0 \wedge a_1 \wedge \cdots \wedge a_r = e\mathbf{A}_r + \boldsymbol{E}\mathbf{A}_r^+. \tag{6.83}$$

On the other hand, the *volume* (or *content*) of the simplex is $k!|\mathbf{A}_r^+|$, where

$$|\mathbf{A}_r^+|^2 = (\mathbf{A}_r^+)^\dagger \mathbf{A}_r^+ = -\big(a_r \wedge \cdots \wedge a_0 \wedge e\big) \cdot \big(e \wedge a_0 \wedge \wedge \cdots \wedge a_r\big),$$
$$= -\left(-\frac{1}{2}\right)^r \begin{vmatrix} 0 & 1 & \cdots & 1 \\ 1 & & & \\ \vdots & & d_{ij}^2 & \\ 1 & & & \end{vmatrix}, \tag{6.84}$$

where $d_{ij} = |\mathbf{a}_i - \mathbf{a}_j|$ is the distance between a pair of points. This determinate is called Cayley–Menger determinant. The directed distance from the origin in $\mathcal{R}^n$ to the plane of the simplex in terms of the points is given by

$$\boldsymbol{d} = \mathbf{A}_r (\mathbf{A}_r^+)^{-1}. \tag{6.85}$$

Thus the square of its absolute value is

$$|\boldsymbol{d}|^2 = \frac{|\mathbf{A}_r|^2}{|\mathbf{A}_r^+|^2} = \frac{(\mathbf{a}_r \wedge \cdots \wedge \mathbf{a}_0) \cdot (\mathbf{a}_0 \wedge \cdots \wedge \mathbf{a}_r)}{(\bar{\mathbf{a}}_r \wedge \cdots \wedge \bar{\mathbf{a}}_1) \cdot (\bar{\mathbf{a}}_1 \wedge \cdots \wedge \bar{\mathbf{a}}_r)}, \tag{6.86}$$

where $\bar{\mathbf{a}}_i = \mathbf{a}_i - \mathbf{a}_0$ for $i = 1, \ldots, r$. In next section, we present the computing of the directed distance in the 3D affine plane.

6.2 The 3D Affine Plane

In the previous section, we described the general properties of the conformal framework. However, sometimes we would like to use only the projective plane of the conformal framework but not the null cone of this space. This will be the case when we use only rigid transformations, and then we will limit ourselves to the *affine plane*, which is an $n+1$-dimensional subspace of the hyperplane of reference $\mathbb{P}(e_\infty, e_0)$.

We have chosen to work in the algebra $G_{4,1}$. Since we deal with homogeneous points, the particular choice of null vectors does not affect the properties of the conformal geometry. Points in the affine plane $\boldsymbol{x} \in \mathbb{R}^{4,1}$ are formed as follows:

$$\boldsymbol{x}^a = \mathbf{x}_e + e_0, \tag{6.87}$$

where $\mathbf{x}_e \in \mathbb{R}^3$. From this equation, we note that e_0 represents the origin (by setting $\mathbf{x}_e = 0$); similarly, e_∞ represents the point at infinity. Then the normalization property is expressed as

$$e_\infty \cdot \boldsymbol{x}^a = -1. \tag{6.88}$$

In this framework, the conformal mapping equation is expressed as

$$\boldsymbol{x}_c = \mathbf{x}_e + \frac{1}{2}\mathbf{x}_e^2 e_\infty + e_0 = \boldsymbol{x}^a + \frac{1}{2}\mathbf{x}_e^2 e_\infty. \tag{6.89}$$

For the case when we will be working on the affine plane exclusively, we will be mainly concerned with a simplified version of the *rejection*. Noting that $\boldsymbol{E} = e_\infty \wedge e_0 = e_\infty \wedge e$, we write an equation for rejection as follows:

$$\begin{aligned} P_E^{\perp}(\boldsymbol{x}_c) &= (\boldsymbol{x}_c \wedge \boldsymbol{E})\boldsymbol{E} = (\boldsymbol{x}_c \wedge \boldsymbol{E}) \cdot \boldsymbol{E} = (e_\infty \wedge e_0) \cdot e_0 + (\boldsymbol{x}_c \wedge e_\infty) \cdot e_0, \\ \mathbf{x}_e &= -e_0 + (\boldsymbol{x}_c \wedge e_\infty) \cdot e_0. \end{aligned} \tag{6.90}$$

Now, since the points in the affine plane have the form $\boldsymbol{x}^a = \boldsymbol{x}_e + e_0$, we conclude that

$$\boldsymbol{x}^a = (\boldsymbol{x}_c \wedge e_\infty) \cdot e_0 \tag{6.91}$$

is the mapping from the horosphere to the affine plane.

6.2.1 Lines and Planes

The lines and planes in the affine plane are expressed in a similar fashion to their conformal counterparts as the *join* of two and three points, respectively:

$$\boldsymbol{L}^a = \boldsymbol{x}_1^a \wedge \boldsymbol{x}_2^a, \tag{6.92}$$

$$\boldsymbol{\Pi}^a = \boldsymbol{x}_1^a \wedge \boldsymbol{x}_2^a \wedge \boldsymbol{x}_3^a. \tag{6.93}$$

Note that unlike their conformal counterparts, the line is a *bivector* and the plane is a *trivector*. As seen earlier, these equations produce a moment-direction representation; thus,

$$\boldsymbol{L}^a = e_\infty \mathbf{d} + \boldsymbol{B}, \tag{6.94}$$

where $\mathbf{d}$ is a vector representing the direction of the line and $\boldsymbol{B}$ is a bivector representing the moment of the line. Similarly, we have that

$$\boldsymbol{\Pi}^a = e_\infty \mathbf{n} + \delta e_{123}, \tag{6.95}$$

where $\mathbf{n}$ is the normal vector to the plane and δ is a scalar representing the distance from the plane to the origin. Note that in any case, the direction and normal can be retrieved with $\mathbf{d} = e_\infty \cdot \boldsymbol{L}^a$ and $\mathbf{n} = e_\infty \cdot \boldsymbol{\Pi}^a$, respectively.

In this framework, the intersection or *meet* has a simple expression, too. Let $\boldsymbol{A}^a = a_1^a \wedge \cdots \wedge a_r^a$ and $\boldsymbol{B}^a = b_1^a \wedge \cdots \wedge b_s^a$. Then the meet is defined as

$$\boldsymbol{A}^a \cap \boldsymbol{B}^a = \boldsymbol{A}^a \cdot (\boldsymbol{B}^a \cdot \bar{I}_{\boldsymbol{A}^a \cup \boldsymbol{B}^a}), \tag{6.96}$$

where $\bar{I}_{\boldsymbol{A}^a \cup \boldsymbol{B}^a}$ is either $e_{12}e_\infty$, $e_{23}e_\infty$, $e_{31}e_\infty$, or $e_{123}e_\infty$, according to which basis vectors span the largest common space of $\boldsymbol{A}^a$ and $\boldsymbol{B}^a$.

6.2.2 Directed Distance

The so-called Hessian normal form is well known from vector analysis to be a convenient representation to specify lines and planes using their distance from the origin (the Hesse distance or directed distance). In this section, we are going to show how CGA can help us to obtain the Hesse distance for more general simplexes and not only for lines and planes. Figure 6.8a, b depicts a line and a plane, respectively, that will help us to develop our equations. Let $\boldsymbol{A}^k$ be a k-line (or plane); then it consists of a *momentum* M^k of degree k and of a *direction* D^{k-1} of degree $k-1$. For instance, given three Euclidean points a_1, a_2, a_3, their 2-simplex defines a dual 3-plane in CGA that can be expressed as

$$\boldsymbol{A}^k \equiv \Phi = M^3 + D^2 e_0 = a_1 \wedge a_2 \wedge a_3 + (a_2 - a_1) \wedge (a_3 - a_1) e_0. \tag{6.97}$$

Then the directed distance of this plane, denoted as $\boldsymbol{p}^k$, can be obtained taking the inner product between the unit direction D_u^{k-1} and the moment M^k. Indeed, from (6.1) and using expressions (6.6), we get the direction from $\Phi \cdot e_\infty = D^{k-1}$ and then its unitary expression D_u^{k-1} dividing D^{k-1} by its magnitude. Schematically,

$$\boldsymbol{A}^k \longrightarrow \boldsymbol{A}^k \cdot e_\infty = \boldsymbol{D}^{k-1} \longrightarrow \boldsymbol{D}_u^{k-1} = \frac{\boldsymbol{D}^{k-1}}{\left|\boldsymbol{D}^{k-1}\right|}. \tag{6.98}$$

Finally, the directed distance $\boldsymbol{p}^k$ of $\boldsymbol{A}^k$ is

$$\boldsymbol{p}^k = \boldsymbol{D}_u^{k-1} \cdot \boldsymbol{A}^k, \tag{6.99}$$

where the dot operation basically takes place between the direction $\boldsymbol{D}_u^{k-1}$ and the momentum of $\boldsymbol{A}^k$. Obviously, the directed distance vector $\boldsymbol{p}^k$ touches orthogonally the k-plane $\boldsymbol{A}^k$, and as we mentioned at the beginning of this subsection, the magnitude $\left|\boldsymbol{p}^k\right|$ equals the Hesse distance. For the sake of simplicity, in Fig. 6.8a, b only $\boldsymbol{D}^{k-1} \cdot \boldsymbol{L}^k$ and $\boldsymbol{D}^{k-1} \cdot \boldsymbol{\Phi}^k$ are, respectively, shown. Now, having this point from the first object, we can use it to compute the directed distance from the k-plane $\boldsymbol{A}^k$ parallel to the object $\boldsymbol{B}^k$ as follows:

$$d[\boldsymbol{A}^k, \boldsymbol{B}^k] = d[\boldsymbol{D}^{k-1} \cdot \boldsymbol{A}^k, \boldsymbol{B}^k] = d[(e_\infty \cdot \boldsymbol{A}^k) \cdot \boldsymbol{A}^k, \boldsymbol{B}^k]. \tag{6.100}$$

6.3 The Lie Algebra

We should introduce the analog of the complex "doubling" bivector for conformal geometry. This corresponds to the Minkowski plane of Eq. (6.7), namely,

$$\boldsymbol{E} = e_i \bar{e}_j = e_+ \wedge e_- = e_+ e_-. \tag{6.101}$$

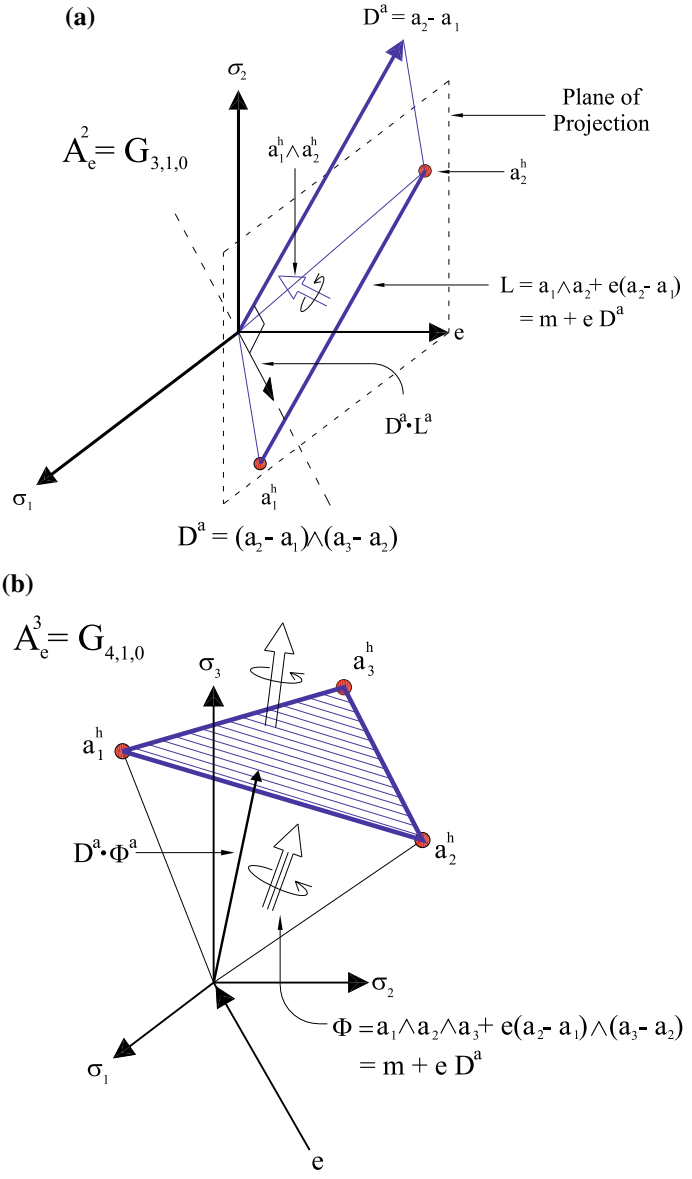


Fig. 6.8 **a** Line in 2D affine space. **b** Plane in the 3D affine space (note that the 3D space is "lifted" by a null vector *e*

Thus, the bivector generators are the set of bivectors that commute with $\boldsymbol{E}$. We proceed similarly as with the unitary group and formulate an algebraic constraint so for any $\boldsymbol{x},\ \boldsymbol{y} \in G_{n+1,1}$

$$[(\boldsymbol{x}\cdot\boldsymbol{E})\wedge(\boldsymbol{y}\cdot\boldsymbol{E})]\times\boldsymbol{E}=\boldsymbol{x}\wedge(\boldsymbol{y}\cdot\boldsymbol{E})+(\boldsymbol{x}\cdot\boldsymbol{E})\wedge y=(\boldsymbol{x}\wedge\boldsymbol{y})\times\boldsymbol{E} \quad (6.102)$$

after a simple algebraic manipulation,

$$[\boldsymbol{x}\wedge\boldsymbol{y}-(\boldsymbol{x}\cdot\boldsymbol{E})(\boldsymbol{y}\cdot\boldsymbol{E})]\times\boldsymbol{E}=0. \quad (6.103)$$

Now, using this constraint, we can again try all combinations of $\{e_i, \bar{e}_i\}$ to produce the bivector basis for the Lie algebra of the conformal group:

$$\boldsymbol{E}_i = e_+e_-, \quad (6.104)$$

$$\boldsymbol{B}_{ij} = e_ie_j \qquad (i<j=1,\ldots,n), \quad (6.105)$$

$$\bar{\boldsymbol{E}}_{ij} = e_ie_\pm \qquad (i=1,\ldots,n). \quad (6.106)$$

In the next section, we will use this set of Lie algebra operators to define the Lie groups of the conformal geometric algebra.

6.4 Conformal Transformations

In the middle of the nineteenth century, J. Liouville proved, for the three-dimensional case, that any conformal mapping on the whole of $\mathbb{R}^n$ can be expressed as a composition of *inversions in spheres* and *reflections in hyperplanes* [8]. In particular, *rotation, translation, dilation*, and *inversion* mappings will be obtained with these two mappings. In conformal geometric algebra, this concepts are simplified, due to the isomorphism between the conformal group on $\mathbb{R}^n$ and the Lorentz group on $\mathbb{R}^{n+1}$, which helps us to express, with a linear Lorentz transformation, a nonlinear conformal transformation, and then to use *versor* representation to simplify the *composition of transformations* with *multiplication of vectors* [1]. Thus, using conformal geometric algebra, it is computationally more efficient and simpler to interpret the geometry of the conformal mappings than with matrix algebra. A transformation of geometric figures is said to be *conformal* if it preserves the *shape* of the figures, that is, whether it preserves the angles and hence the shapes of straight lines and circles. In particular, rotation and translation mappings are conformal and are also called *direct-motion* transformations. Inversion and reflection mappings preserve the magnitude of the angle but reverse its direction; they are also called *opposite-motion* transformations.

Any conformal transformation in $\mathbb{R}^n$,

$$\boldsymbol{x} \longmapsto \boldsymbol{x}\frac{1}{1+\boldsymbol{a}\boldsymbol{x}}, \tag{6.107}$$

can be expressed as a composite of inversions and a translation:

$$\boldsymbol{x} \overset{\longmapsto}{inversion} \quad \frac{\boldsymbol{x}}{\boldsymbol{x}^2}, \tag{6.108}$$

$$\overset{\longmapsto}{translation} \quad \frac{\boldsymbol{x}}{\boldsymbol{x}^2}+\boldsymbol{a}, \tag{6.109}$$

$$\overset{\longmapsto}{inversion} \quad \frac{\frac{\boldsymbol{x}}{\boldsymbol{x}^2}+\boldsymbol{a}}{(\frac{\boldsymbol{x}}{\boldsymbol{x}^2}+\boldsymbol{a})(\frac{\boldsymbol{x}}{\boldsymbol{x}^2}+\boldsymbol{a})} = \boldsymbol{x}\frac{1}{1+\boldsymbol{a}\boldsymbol{x}}. \tag{6.110}$$

The conformal transformation in conformal geometric algebra uses a versor representation,

$$g(\boldsymbol{x}_c) = \boldsymbol{G}\boldsymbol{x}_c(\boldsymbol{G}^*)^{-1} = \sigma\boldsymbol{x}'_c, \tag{6.111}$$

where $\boldsymbol{x}_c \in \mathbb{R}^{n+1,1}$, $\boldsymbol{G}$ is a versor, and σ is a scalar. $\boldsymbol{G}$ can be expressed in CGA as a composite of versors for transversion, translation, and rotation as follows:

$$\boldsymbol{G} = \boldsymbol{K}_{\mathbf{b}}\boldsymbol{T}_{\mathbf{a}}\boldsymbol{R}_{\alpha}. \tag{6.112}$$

These individual versors will be explained next.

6.4.1 Inversion

By the classical definition, an *inversion T_S with respect to a sphere S* (of radius ρ and center $\mathbf{c}$) is such that for any point q at a distance d from $\mathbf{c}$, $T_S(q)$ will be in the same ray from $\mathbf{c}$ to q and at a distance ρ^2/d from $\mathbf{c}$, see Fig. 6.9a. We comment some of the main properties of this transformation: (a) the inverse of a plane through the center of inversion is the plane itself; (b) the inverse of a plane not passing through the center of inversion is a sphere passing through the center of inversion; (c) the inverse of a sphere through the center of inversion is a plane not passing through the center of inversion; (d) the inverse of a sphere not passing through the center of inversion is a sphere not passing through the center of inversion, see Fig. 6.9b; and (e) the inversion in a sphere maps lines and circles to lines and circles.

In the context of the conformal geometry, the general form of a reflection about a vector is

$$s(\boldsymbol{x}_c) = -\boldsymbol{s}\boldsymbol{x}_c\boldsymbol{s}^{-1} = \boldsymbol{x}_c - 2(\boldsymbol{s}\cdot\boldsymbol{x}_c)\boldsymbol{s}^{-1} = \sigma\boldsymbol{x}'_c, \tag{6.113}$$

where $\boldsymbol{s}\boldsymbol{x}+\boldsymbol{x}\boldsymbol{s} = 2(\boldsymbol{s}\cdot\boldsymbol{x})$, from the definition of the Clifford product between two vectors. We will now analyze what happens when s represents a sphere. Recall that

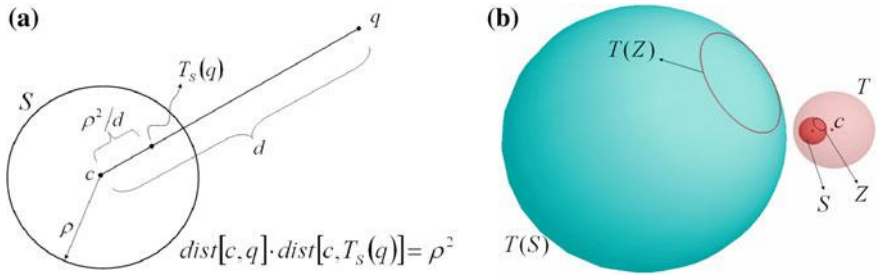


Fig. 6.9 **a** The point $T_S(q)$ is the inverse point of q with respect to the circle S, and vice versa. **b** The inversion with respect to a sphere T mapping sphere to sphere and circle to circle, if they are not passing through the center of T

the equation of a sphere of radius ρ centered at point $\boldsymbol{c}_c$ is the vector

$$\boldsymbol{s} = \boldsymbol{c}_c - \frac{1}{2}\rho^2 e_\infty. \tag{6.114}$$

If $\boldsymbol{s}$ represents the unit sphere centered at the origin, then $\boldsymbol{s}$ and $\boldsymbol{s}^{-1}$ reduce to $e_0 - \frac{1}{2}e_\infty$. Hence, $-2(\boldsymbol{s}\cdot\boldsymbol{x}_c) = \mathbf{x}_e^2 - 1$, and Eq. (6.113) becomes

$$\sigma\boldsymbol{x}_c' = \left(\mathbf{x}_e + \frac{1}{2}\mathbf{x}_e^2 + e_\infty\right) + \left(\mathbf{x}_e^2 - 1\right)(e_0 - e_\infty) = \mathbf{x}_e^2\left(\mathbf{x}_e^{-1} + \mathbf{x}_e^{-2}e_\infty + e_0\right), \tag{6.115}$$

which is the conformal mapping for $\mathbf{x}_c^{-1}$.

To see how a general sphere inverts a point, we return to Eq. (6.114) to get

$$\boldsymbol{s}\cdot\boldsymbol{x}_c = \boldsymbol{c}_c\cdot\boldsymbol{x}_c - \rho^2 e_\infty\cdot\boldsymbol{x}_c = -\left[(\mathbf{x}_e - \mathbf{c}_e)^2 - \rho^2\right]. \tag{6.116}$$

Inserting Eq. (6.115) and a little algebra gives

$$\sigma\boldsymbol{x}_c' = \left(\frac{\mathbf{x}_e - \mathbf{c}_e}{\rho}\right)^2\left[g(\boldsymbol{x}_c) + \frac{1}{2}g^2(\mathbf{x}_c)e_\infty + e_0\right], \tag{6.117}$$

where

$$g(\mathbf{x}_e) = \frac{\rho^2}{\mathbf{x}_e - \mathbf{c}_e} + \mathbf{c}_e = \frac{\rho^2(\mathbf{x}_e - \mathbf{c}_e)}{(\mathbf{x}_e - \mathbf{c}_e)^2} + \mathbf{c}_e \tag{6.118}$$

is the inversion in $\mathbb{R}^n$.

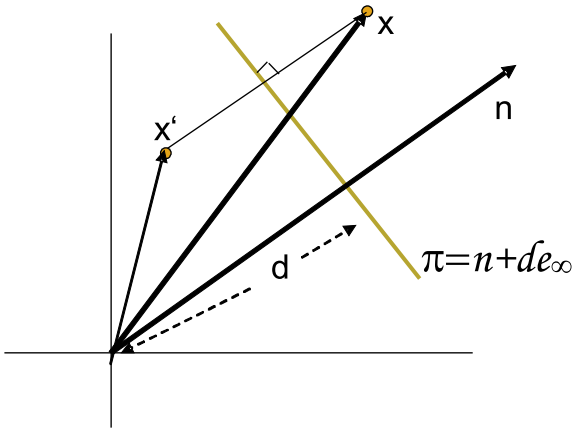


Fig. 6.10 Reflection of a point x with respect to the plane π

6.4.2 Reflection

The reflection of conformal geometric entities helps us to do any other transformation. The reflection of a point x with respect to the plane π is equal to x minus twice the direct distance between the point and plane (see Fig. 6.10), that is, $x = x - 2(\pi \cdot x)\pi^{-1}$. To simplify this expression, recall the property of Clifford product of vectors $2(b \cdot a) = ab + ba$.

The reflection could be written as

$$x' = x - (\pi x - x\pi)\pi^{-1}, \tag{6.119}$$

$$x' = x - \pi x \pi^{-1} - x\pi\pi^{-1}, \tag{6.120}$$

$$x' = -\pi x \pi^{-1}. \tag{6.121}$$

For any geometric entity Q, the reflection with respect to the plane π is given by

$$Q' = \pi Q \pi^{-1}. \tag{6.122}$$

6.4.3 Translation

The translation of conformal entities can be obtained by carrying out two reflections in parallel planes π_1 and π_2 (see Fig. 6.11):

$$Q' = T_a \underbrace{(\pi_2 \pi_1)} Q \widetilde{T}_a \underbrace{(\pi_1^{-1} \pi_2^{-1})}, \tag{6.123}$$

$$T_a = (n + de_\infty)n = 1 + \frac{1}{2} a e_\infty = e^{-\frac{a}{2} e_\infty}, \tag{6.124}$$

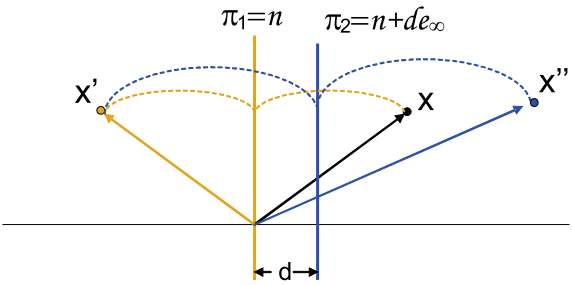


Fig. 6.11 Reflection about parallel planes

where $a = 2dn$.

Since $\mathbf{a}e_\infty = b^i e_i(e_+ + e_-)$, the Lie algebra generators or bivectors for the translator T_a are $e_i e_+$ and $e_i e_-$ (see Eq. (6.106)).

6.4.4 Transversion

A transversion can be generated from two inversions and a translation. A transversor has the form

$$\boldsymbol{K}_\mathbf{b} = e_+ \boldsymbol{T}_\mathbf{b} e_+ = (e_\infty - e_0)(1 + \mathbf{b}e_\infty)(e_\infty - e_0) = 1 + \mathbf{b}e_0. \tag{6.125}$$

The transversion generated by $K_\mathbf{b}$ can be expressed in various forms:

$$g(\mathbf{x}_e) = \frac{\mathbf{x}_e - \mathbf{x}_e^2\mathbf{b}}{1 - 2\mathbf{b}\cdot\mathbf{x}_e + \mathbf{x}_e^2\mathbf{b}^2} = \mathbf{x}_e(1 - \mathbf{b}\mathbf{x}_e)^{-1} = (\mathbf{x}_e^{-1} - \mathbf{b})^{-1}. \tag{6.126}$$

The last form can be written down directly as an inversion followed by a translation and another inversion. Note that a transversion uses as a null vector the origin e_0, whereas the translator uses the point at infinity e_∞. Since $\mathbf{b}e_0 = b^i e_i \frac{1}{2}(e_- - e_+)$, the Lie algebra generators or bivectors for the transversion T_a are $e_i e_+$ and $e_i e_-$ (see Eq. (6.106)). In fact, the bivector basis is the same for the translator and for the transversion.

6.4.5 Rotation

A rotation is the product of two reflections with respect to two nonparallel planes (see Fig. 6.12):

$$Q' = R_\theta\underbrace{(\pi_2\pi_1)}Q\widetilde{R}_\theta\underbrace{(\pi_1^{-1}\pi_2^{-1})}, \tag{6.127}$$

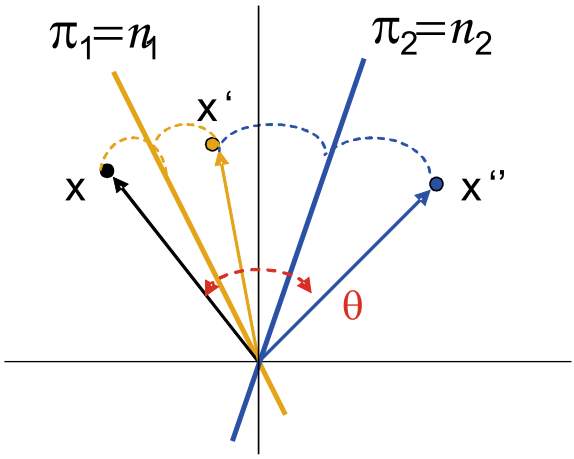


Fig. 6.12 Reflection about nonparallel planes

or, computing the conformal product of the normals of the planes,

$$R_\theta = n_2 n_1 = \cos\left(\frac{\theta}{2}\right) - \sin\left(\frac{\theta}{2}\right) l = e^{-\frac{\theta}{2} l}. \tag{6.128}$$

There are three Lie algebra generators or bivectors for the rotors; they are of the type $B_{ij} = e_i e_j$ (see Eq. (6.105)).

With $l = n_2 \wedge n_1$, and θ twice the angle between the planes π_2 and π_1, the screw motion called motor related to an arbitrary axis L is $M = T R \widetilde{T}$:

$$Q' = M_\theta \underbrace{(T R \widetilde{T})} Q \widetilde{M}_\theta \underbrace{((T \widetilde{R} \widetilde{T}))}, \tag{6.129}$$

$$M_\theta = T R \widetilde{T} = \cos\left(\frac{\theta}{2}\right) - \sin\left(\frac{\theta}{2}\right) L = e^{-\frac{\theta}{2} L}. \tag{6.130}$$

The direct kinematics for serial robot arms can be expressed as a succession of motors and surprisingly is valid for points, lines, planes, circles, and spheres.

$$Q' = \prod_{i=1}^{n} M_i \, Q \prod_{i=1}^{n} \widetilde{M}_{n-i+1}. \tag{6.131}$$

6.4.6 *Rigid Motion Using Flags*

In Sect. 5.10.1, the concept of *flag* or *soma* is explained. This helps to embed the frame in a single algebraic as depicted in Fig. 6.13. The flag includes the representations of a point, line, and a plane, all entities with a common reference point

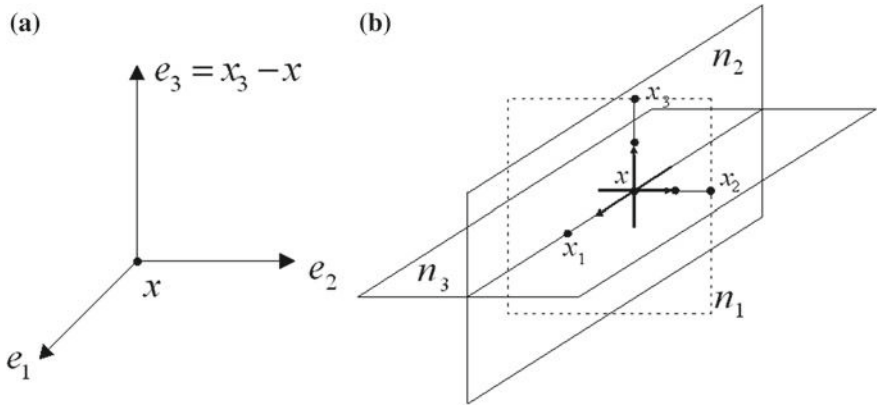


Fig. 6.13 **a** Flag: point, line, and plane with a common point

$$\boldsymbol{F} = \boldsymbol{x} + \boldsymbol{L}^* + \pi^* = \boldsymbol{x} + I_c \boldsymbol{Q}, \tag{6.132}$$

where

$$\begin{aligned} \boldsymbol{L}^* &= \boldsymbol{x} \wedge \boldsymbol{x}_1 \wedge e_\infty = \boldsymbol{x} \wedge (\boldsymbol{x}_1 - \boldsymbol{x}) \wedge e_\infty = \boldsymbol{x} \wedge e_1 \wedge e_\infty = I_c \boldsymbol{n}_2 \boldsymbol{n}_3 \\ \pi^* &= \boldsymbol{x} \wedge \boldsymbol{x}_1 \wedge \boldsymbol{x}_2 \wedge e_\infty = \boldsymbol{x} \wedge e_1 \wedge e_2 \wedge e_\infty = e_2 \wedge \boldsymbol{L}^* = I_c \boldsymbol{n}_3. \end{aligned}$$

Combining these results, one gets $\boldsymbol{Q} = \boldsymbol{n}_3 + \boldsymbol{n}_2 \boldsymbol{n}_3 = (1 + \boldsymbol{n}_2)\boldsymbol{n}_3$ which is a conic, namely, it fulfills

$$\boldsymbol{x} \wedge \boldsymbol{F} = \boldsymbol{x} \wedge (\boldsymbol{L}^* + \pi^*) = \boldsymbol{x} \wedge (I_c \boldsymbol{Q}) = 0. \tag{6.133}$$

Its dual expression is

$$\boldsymbol{x} \cdot \boldsymbol{Q} = 0, \tag{6.134}$$

where

$$\boldsymbol{Q}^2 = (\boldsymbol{n}_3 + \boldsymbol{n}_2 \boldsymbol{n}_3)(\boldsymbol{n}_3 - \boldsymbol{n}_2 \boldsymbol{n}_3) = 0. \tag{6.135}$$

The previous equations describe a point lying on an absolute conic. Recall in projective geometry the equation of homogeneous points lying on the absolute conic.

Finally, the rigid body can be nicely modeled using the flag $\boldsymbol{F}$ as follows, given a motor $\boldsymbol{M} = \boldsymbol{T}\boldsymbol{R}$:

$$\boldsymbol{F}' = \boldsymbol{M}\boldsymbol{F}\widetilde{\boldsymbol{M}} = \boldsymbol{M}\boldsymbol{x}\widetilde{\boldsymbol{M}} + \boldsymbol{M}\boldsymbol{L}^*\widetilde{\boldsymbol{M}} + \boldsymbol{M}\pi^*\widetilde{\boldsymbol{M}}. \tag{6.136}$$

Here, this linearity is due to the congruence theorem.

6.4.7 Dilation

A dilation is the composite of two successive inversions centered at the origin. Using the unit sphere $s_1 = e_0 - \frac{1}{2}e_\infty$, another sphere of arbitrary radius ρ, and $s_2 = e_0 - \frac{1}{2}\rho^2 e_\infty$ as inversors, we get

$$(e_0 - e_\infty)(e_0 - \rho^2 e_\infty) = (1 - \boldsymbol{E}) + (1 + \boldsymbol{E})\rho^2. \tag{6.137}$$

Normalizing to unity, we have

$$\boldsymbol{D}_\rho = (1 + \boldsymbol{E})\rho + (1 - \boldsymbol{E})\rho^{-1} = e^{\boldsymbol{E}\phi}, \tag{6.138}$$

where $\phi = \ln \rho$. To prove that this is indeed a dilation, we note that

$$\boldsymbol{D}_\rho e_\infty \boldsymbol{D}_\rho^{-1} = \rho^{-2} e, \tag{6.139}$$

and similarly,

$$\boldsymbol{D}_\rho e_0 \boldsymbol{D}_\rho^{-1} = \rho^2 e_0. \tag{6.140}$$

Therefore,

$$\boldsymbol{D}_\rho(\mathbf{x}_e + \mathbf{x}_e^2 e_\infty + e_0)\boldsymbol{D}_\rho^{-1} = \rho^2[\rho^{-2}\mathbf{x}_e + (\rho^{-2}\mathbf{x}_e)^2 e_\infty + e_0], \tag{6.141}$$

which is the conformal mapping $g(\boldsymbol{x}_e) = \sigma \boldsymbol{x}'_c$ with $\boldsymbol{x}'_c = \boldsymbol{x}'_e + \boldsymbol{x}'^2 e_\infty + e_0$, where $\boldsymbol{x}'_e = \rho^{-2}\mathbf{x}_e$. The Lie algebra generator or bivector for the dilations is $E = e_+ e_-$ or the Minkowski plane, see Eq. (6.104).

6.4.8 Involution

The bivector E (the Minkowski plane of $\mathbb{R}^{1,1}$) represents an operation that corresponds to the main involution, but for an r-blade $\boldsymbol{A}_r$, namely, $\bar{\boldsymbol{A}}_r = (-1)^r \boldsymbol{A}_r$. In particular, for vectors $\bar{\mathbf{x}}_e = -\boldsymbol{x}_e$, which can be easily obtained by applying the versor E:

$$\boldsymbol{E}(\mathbf{x}_e + \mathbf{x}_e^2 e_\infty + e_0)\boldsymbol{E} = -\left(-\mathbf{x}_e + \frac{1}{2}\mathbf{x}_e^2 e_\infty + e_0\right). \tag{6.142}$$

This expression corresponds to the conformal mapping of $-\boldsymbol{x}_e$, thus confirming that the versor E represents the main involution for $\mathbb{R}^n$. This means that the main involution is a reflection via the Minkowski plane $\mathbb{R}^{1,1}$.

6.4.9 Conformal Transformation

Finally, using previous results, we can now write a canonical decomposition of a conformal transformation in terms of individual versors:

$$\boldsymbol{G} = \boldsymbol{K}_{\mathbf{b}} \boldsymbol{T}_{\mathbf{a}} \boldsymbol{R}_{\alpha}, \tag{6.143}$$

where the versors are as follows: for transversion $\boldsymbol{K}_{\mathbf{b}} = e_+ \boldsymbol{T}_{\mathbf{b}} e_+ = 1 + \mathbf{b} e_0$, for translation $\boldsymbol{T}_{\mathbf{a}} = 1 + \frac{1}{2} a e_\infty$, and for rotation $\boldsymbol{R}_{\alpha}$. This decomposition reveals the structure of the three-parameter group $\{G \in \mathbb{R}_{1,1} | G^* G^\dagger = 1\} \simeq GL_2(\mathbb{R})$. Note that in conformal geometric algebra the conformal transformation is built by successive multiplicative versors: in contrast to $\mathbb{R}^n$, the conformal transformation of Eq. (6.110) is highly nonlinear, and its application will require of complicated nonlinear and slow algorithms.

6.5 Ruled Surfaces

Conics, ellipsoids, helicoids, and hyperboloids of one sheet are entities that cannot be directly described in CGA, but can be modeled with its multivectors. In particular, a ruled surface is a surface generated by the displacement of a straight line (called *generatrix*) along a directing curve or curves (called a *directrices*). The plane is the simplest ruled surface, but now we are interested in nonlinear surfaces generated as ruled surfaces. For example, a circular cone is a surface generated by a straight line through a fixed point and a point in a circle. It is well known that the intersection of a plane with the cone can generate the conics (see Fig. 6.14). In [9], the cycloidal curves can be generated by two coupled twists. In this section, we are going to see how these and other curves and surfaces can be obtained using only multivectors of CGA.

6.5.1 Cone and Conics

A circular cone is described by a fixed point v_0 (*vertex*), a dual circle $z_0 = a_0 \wedge a_1 \wedge a_2$ (*directrix*), and a rotor $\mathbf{R}(\theta, l)$, $\theta \in [0, 2\pi)$ rotating the straight line $L(v_0, a_0) = v_0 \wedge a_0 \wedge e_\infty$ (*generatrix*) along the axis of the cone $l_0 = z_0 \cdot e_\infty$. Then, the cone w is generated as

$$w = \boldsymbol{R}(\theta, l_0)\, L(v_0, a_0) \widetilde{\boldsymbol{R}}(\theta, l_0)\,, \quad \theta \in [0, 2\pi)\,. \tag{6.144}$$

A conic curve can be obtained with the *meet* of a cone and a plane, see Fig. 6.14.

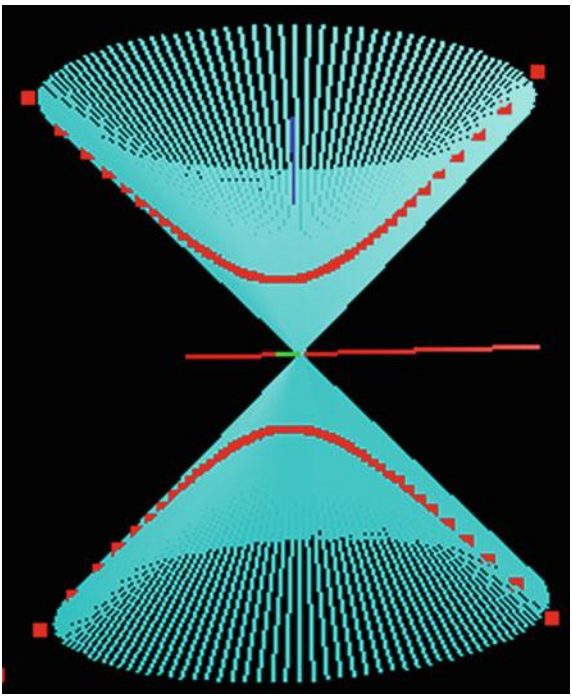


Fig. 6.14 Hyperbola as the *meet* of a cone and a plane

6.5.2 Cycloidal Curves

The family of the cycloidal curves can be generated by the rotation and translation of one or two circles. For example (see Fig. 6.15), the cycloidal family of curves generated by two circles of radius r_0 and r_1 is expressed by the following motor:

$$\boldsymbol{M} = \boldsymbol{T}\boldsymbol{R}_1\widetilde{\boldsymbol{T}}\boldsymbol{R}_2, \tag{6.145}$$

where

$$\boldsymbol{T} = \boldsymbol{T}((r_0 + r_1)(\sin(\theta)e_1 + \cos(\theta)e_2)), \tag{6.146}$$

$$\boldsymbol{R}_1 = \boldsymbol{R}_1\left(\frac{r_0}{r_1}\theta\right), \tag{6.147}$$

$$\boldsymbol{R}_2 = \boldsymbol{R}_2(\theta). \tag{6.148}$$

Then, each conformal point x is transformed as $\boldsymbol{M}x\widetilde{\boldsymbol{M}}$.

Fig. 6.15 A laser welding following a 3D curve: the projection of a cycloidal curve over a sphere

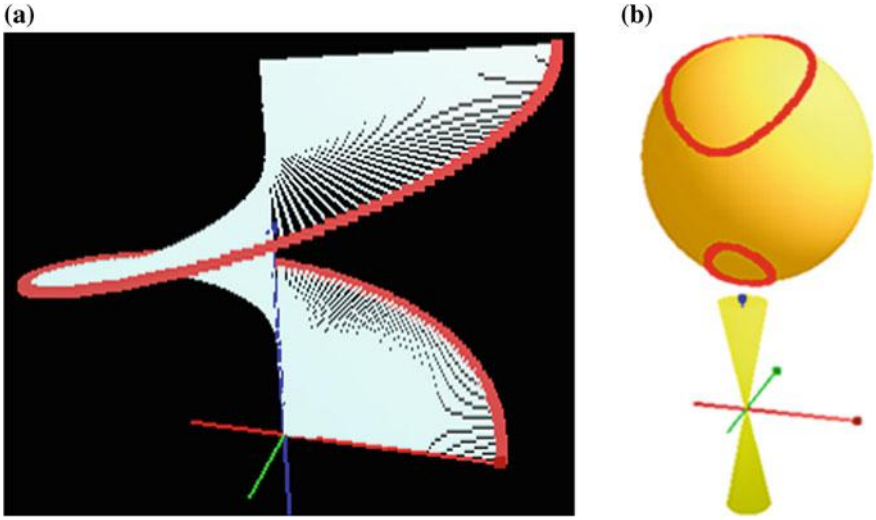


Fig. 6.16 **a** The helicoid is generated by the rotation and translation of a line segment. In CGA, the *motor* is the desired multivector. **b** Intersection as the *meet* of a sphere and a cone

6.5.3 Helicoid

We can obtain the ruled surface called a helicoid rotating a ray segment in a similar way as the spiral of Archimedes. So, if the axis e_3 is the directrix of the rays and is orthogonal to them, then the translator that we need to apply is a multiple of θ, the angle of rotation (see Fig. 6.16a).

6.5.4 Sphere and Cone

Let us see an example of how the use of algebra of incidence in CGA simplifies the algebraic formulation. The intersection of a cone and a sphere in general position, that is, the axis of the cone does not pass through the center of the sphere, is the three-dimensional curve of all Euclidean points (x, y, z) such that x and y satisfy the quadratic equation

$$\left[x^2\left(1+\frac{1}{c^2}\right) - 2x_0x + y^2\left(1+\frac{1}{c^2}\right) - 2y_0y + x_0^2 + y_0^2 + z_0^2 - r^2\right]^2 = 4z_0^2(x^2+y^2)/c^2 \quad (6.149)$$

and x, y, and z the quadratic equation

$$(x-x_0)^2 + (y-y_0)^2 + (z-z_0)^2 = r^2 \quad (6.150)$$

(see Fig. 6.16b). In CGA, the set of points q of the intersection can be expressed as the meet of the dual sphere s and the cone w, Eq. (6.144), defined in terms of its generatrix L, that is,

$$q = (s^*) \cdot [\boldsymbol{R}\,(\theta, l_0)\, L(v_0, a_0)\widetilde{\boldsymbol{R}}\,(\theta, l_0)],\ \ \theta \in [0, 2\pi)\,. \quad (6.151)$$

Thus, in CGA, we only need (6.151) to express the intersection of a sphere and a cone; meanwhile, in Euclidean geometry it is necessary to use the complicated Eqs. (6.149) and (6.150)).

6.5.5 Hyperboloid, Ellipsoids, and Conoid

The rotation of a line over a circle can generate a hyperboloid of one sheet (see Fig. 6.17a). The ellipse is a curve of the family, of the cycloid; we can obtain an ellipsoid with a translator and a dilator. The cylindroid or Plücker conoid is a ruled surface (see Fig. 6.17b). This ruled surface is like the helicoid, where the magnitude of the translator parallel to the e_3 axis is a multiple of $\cos(\theta)\sin(\theta)$.

6.6 Exercises

6.1 Formulate the equation of a sphere using conformal geometric algebra.
6.2 Derive the representation of the point pair (0-sphere) $\boldsymbol{P}\,\boldsymbol{P}$ spanned by the points $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ at locations e_1 and e_2 with weights 3 and -2.
6.3 Compute the center and radius of $\boldsymbol{P}\,\boldsymbol{P}$.

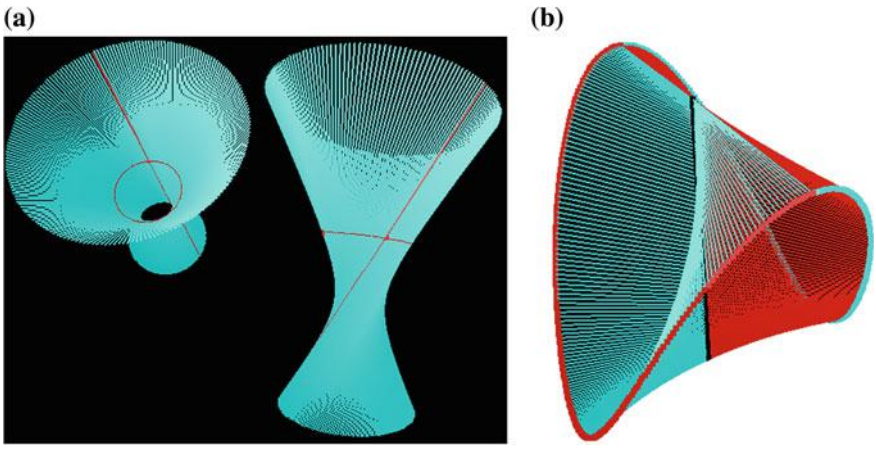


Fig. 6.17 **a** Hyperboloid as the rotor of a line. **b** The Plücker conoid as a ruled surface

6.4 Give the dual representation of $\boldsymbol{P}\,\boldsymbol{P}$ and compute its center and radius.

6.5 Give the IPNS representation of the circle $\boldsymbol{z}$ through the points $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ and the unit point e_1. Compute its square radius and center, and the orientation vector of the circle $\boldsymbol{z}$ passing its center.

6.6 Compute the IPNS representation of the sphere $\boldsymbol{s}$ through $\boldsymbol{z}$ and the origin.

6.7 Compute the OPNS representation of the sphere $\boldsymbol{s}^*$ and read off its center and square radius directly from that dual representation.

6.8 In conformal geometric algebra, write the equations of the direct distances between a point and a line, a point and a plane, and a line and a plane.

6.9 Compute the reflexion of the line $\boldsymbol{L}$ which crosses the points e_1 and e_2 with respect to the unitary sphere which is at the origin.

6.10 The dual circle is the intersection of a plane with a sphere, namely, $\boldsymbol{s} \wedge \phi$. Recall that the dual sphere at origin is $\boldsymbol{s}^* = (e_0 - \frac{\rho^2 e_\infty}{2})$. Use CLUCAL to compute the meet of two intersecting dual circles $\boldsymbol{z}_1 = \boldsymbol{T}_{e_1}\big((e_0 - \frac{e_\infty}{2})(-e_3)\big)$ and $\boldsymbol{z}_2 = \boldsymbol{T}_{e_2}\big((e_0 - \frac{e_\infty}{2})(-e_3)\big)$ lying on the $e_1 \wedge e_2$-plane. Here $\boldsymbol{T}_{e_1}$ and $\boldsymbol{T}_{e_2}$ are the translators of the circles along e_1 and e_2, respectively.

6.11 Explain yourself Eq. 6.68

$$(-1)^{\epsilon}\boldsymbol{s}^* \cap \boldsymbol{A}_{r+1} = (\boldsymbol{s}^* I) \cdot \boldsymbol{A}_{r+1} = \boldsymbol{s}\boldsymbol{A}_{r+1} = \boldsymbol{A}_r.$$

This can be used to formulate a sphere in terms of a circle $\boldsymbol{z}$ and a plane π, i.e.,

$$\boldsymbol{s} = \boldsymbol{z}\pi^{-1}.$$

Using geometric entities like, lines, planes, circles, and spheres, draw and compute the projection of a blade A to a blade B, and interpret your resulting blade.

6.12 Prove the flag

$$\boldsymbol{F} = \boldsymbol{x} + \boldsymbol{L}^* + \pi^* = \boldsymbol{x} + I_c \boldsymbol{Q}.$$

6.13 In projective geometry, one applies the homography $H \in \mathbb{P}^3$

$$\mathtt{H} = \begin{bmatrix} \mathtt{R} & \mathbf{t} \\ \mathbf{f} & h \end{bmatrix}$$

to the origin and point at infinity. In conformal geometric algebra, rotate with a rotor $\boldsymbol{R}$, the null vectors, the origin e_o, and the point at infinity e_∞, and explain your results.

6.14 In conformal geometric algebra, translate with a translator $\boldsymbol{T}$, the null vectors, the origin e_o, and the point at infinity e_∞, and explain your results.

6.15 In conformal geometric algebra, write the equations of the direct distances between a sphere and a point, a sphere and a line, and a sphere and a plane.

6.16 In conformal geometric algebra, write the equations of the direct distances between a circle and a point, a circle and a line, a circle and a plane, and a circle and a sphere.

6.17 In conformal geometric algebra, write the equations of the direct distances between a circle and a point, a circle and a line, a circle and a plane, and a circle and a sphere.

6.18 In conformal geometric algebra, given three points lying on a circle, using the conformal split compute the radio and center of the circle.

6.19 In conformal geometric algebra, given three points a_1, a_2, and s_3 in general position using CLICAL [10], eClifford [11], or CLUCAL [12], compute the expanded form of the plane crossing these tree points.

6.20 Theorem proving: Prove in conformal geometric algebra the theorem of Desargues configuration. Recall the 3D projective plane Π^3. Consider that x_1, x_2, x_3 and y_1, y_2, y_3 are the vertices of two triangles in Π^3 and suppose that $(x_1 \wedge x_2) \cap (y_1 \wedge y_2) = z_3$, $(x_2 \wedge x_3) \cap (y_2 \wedge y_3) = z_1$, and $(x_3 \wedge x_1) \cap (y_3 \wedge y_1) = z_2$. You can claim that $c_1 \wedge c_2 \wedge c_3 = 0$ if and only if there is a point p such that $x_1 \wedge y_1 \wedge p = 0 = x_2 \wedge y_2 \wedge p = x_3 \wedge y_3 \wedge p$. (*Hint*: Express the point as linear combinations of a_1, b_1, a_2, b_2, and a_3, b_3. The other half of the proof follows by duality of the classical projective geometry.)

6.21 Theorem proving: Using conformal geometric algebra, prove the Simpson rule using the join of three projected points, see Fig. 6.18. If the projecting point **D** lies at the circumference, the join of the projected points is zero. Take three arbitrary points lying on a unit circumference and a fourth one nearby. Use CLICAL [10], eClifford [11], or CLUCAL [12] for your computations. (*Hint*: First compute the projected points as the meet of three lines passing by the point **D** orthogonal to the triangle sides. The triangle is formed by three arbitrary points lying on the circumference. **A**, **B**, **C**.

6.22 Prove Apollonius's theorem using the incidence algebra in conformal geometric algebra, see Fig. 6.19.

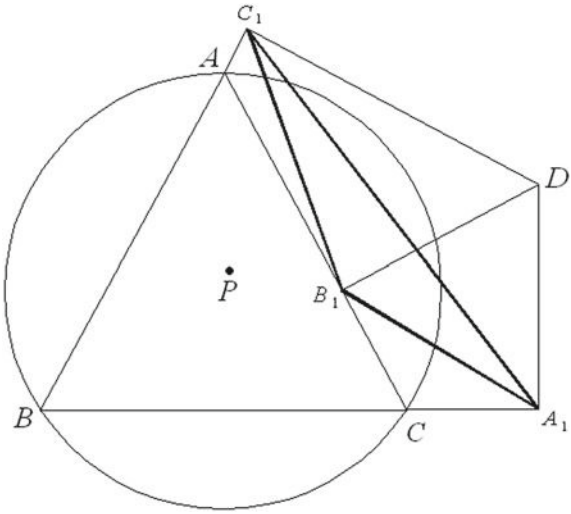


Fig. 6.18 Simson's theorem

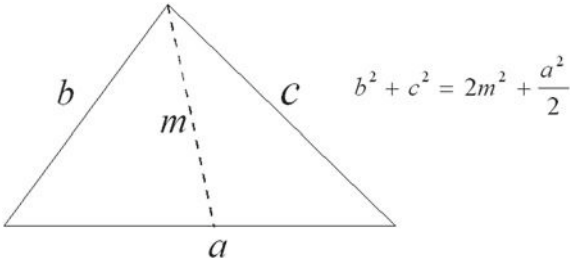


Fig. 6.19 Apollonius's theorem

6.23 Prove the Pascal theorem using the incidence algebra in conformal geometric algebra. Given six points lying in a conic, compute six intersecting lines using the join operation. The meet of these lines should give three intersecting points, and the join of these three intersecting points should be zero. Use CLICAL [10], eClifford [11], or CLUCAL [12] and any six points belonging to a conic of your choice.

6.24 Prove the Desargues theorem using the algebra of incidence in conformal geometric algebra. Let $\boldsymbol{x}_1$, $\boldsymbol{x}_2$, $\boldsymbol{x}_3$ and $\boldsymbol{y}_1$, $\boldsymbol{y}_2$, $\boldsymbol{y}_3$ be the vertices of two triangles in P^2, and suppose that the meets of the lines fulfill the equations $(\boldsymbol{x}_1 \wedge \boldsymbol{x}_2) \cap (\boldsymbol{y}_1 \wedge \boldsymbol{y}_2) = \boldsymbol{z}_3$, $(\boldsymbol{x}_2 \wedge \boldsymbol{x}_3) \cap (\boldsymbol{y}_2 \wedge \boldsymbol{y}_3) = \boldsymbol{z}_1$, and $(\boldsymbol{x}_3 \wedge \boldsymbol{x}_1) \cap (\boldsymbol{y}_3 \wedge \boldsymbol{y}_1) = \boldsymbol{z}_2$. Then the join of these points $\boldsymbol{z}_1 \wedge \boldsymbol{z}_2 \boldsymbol{z}_3 = 0$ if and only if a point $\boldsymbol{p}$ exists such that $\boldsymbol{x}_1 \wedge \boldsymbol{y}_1 \wedge \boldsymbol{p} = \boldsymbol{x}_2 \wedge \boldsymbol{y}_2 \wedge \boldsymbol{p} = \boldsymbol{x}_3 \wedge \boldsymbol{y}_3 \wedge \boldsymbol{p} = 0$. In this problem, use CLICAL [10], eClifford [11], or CLUCAL [12] and define two triangles of your choice.

6.25 A point $\boldsymbol{p}$ lies on the circumcircle of an arbitrary triangle with vertices $\boldsymbol{x}_1$, $\boldsymbol{y}_1$, and $\boldsymbol{z}_1$. From the point $\boldsymbol{p}$, draw three perpendiculars to the three sides of the triangle to meet the circle at points $\boldsymbol{x}_2$, $\boldsymbol{y}_2$, and $\boldsymbol{z}_2$, respectively. Using incidence

algebra in conformal geometric algebra, show that the lines $\boldsymbol{x}_1 \wedge \boldsymbol{x}_2$, $\boldsymbol{y}_1 \wedge \boldsymbol{y}_2$, and $\boldsymbol{z}_1 \wedge \boldsymbol{z}_2$ are parallel.

6.26 In Exercise 5.19, we introduced the barycentric coordinates using a homogeneous model. For this exercise, formulate the expressions for the barycentric coordinates in terms of conformal points.

6.27 Simplex: Show that the weight of $\boldsymbol{x} \wedge \boldsymbol{y} \wedge \boldsymbol{z}/2$ is the area of a triangle $\boldsymbol{xyz}$, and that the weight of $\boldsymbol{x} \wedge \boldsymbol{y} \wedge \boldsymbol{z} \wedge \boldsymbol{z}/3$ is the volume of the tetrahedron $\boldsymbol{xyzz}$.

6.28 The weight of a dual sphere $\boldsymbol{s}^*$ is the weight of its center and equals to $e_\infty \cdot \boldsymbol{s}^*$. Take the dual of this expression to find out that when a sphere passing through the points $\boldsymbol{x}$, $\boldsymbol{y}$, $\boldsymbol{z}$, $\boldsymbol{z}$ becomes zero.

6.29 Prove that the distance measure between a point $\boldsymbol{p}_i$ and the sphere/plane $\boldsymbol{s} = \mathbf{s} + s_4 e_\infty + s_5 e_0$ can be computed in conformal geometry as follows: $\boldsymbol{p}_i \cdot \boldsymbol{s} = \mathbf{p}_i \cdot \mathbf{s} - s_4 - \frac{1}{2} s_5 \mathbf{p}_i^2$ or $\boldsymbol{p}_i \cdot \boldsymbol{s} = \sum_{j=1}^{5} w_{i,j} s_j$, with $w_{i,j} = x_{i,j}$ $j \in \{1, 2, 3\}$, $w_4 = -1$, and $w_5 = -\frac{1}{2}\mathbf{p}_i^2$.

6.30 Use the results of Exercise 6.15 for the sphere to propose the structure of a conformal radial basis function network for data clustering.

6.31 Compute the reflection of the line $\boldsymbol{L}$ through the locations e_1 and e_2 with respect to the unit sphere at the origin.

6.32 Following Exercise 6.30, factorize the result to determine its center and squared radius.

6.33 Compute the reflection of the vector $e_1 + e_2$ in the direction of $3e_3$ with respect to the unit sphere at the origin. Notice the weight in your result.

6.34 Compute the reflection of the line $\boldsymbol{L}$ with respect to the origin.

6.35 Compute the reflection of the line $\boldsymbol{L}$ with respect to the point e_2.

References

1. Li, H., Hestenes, D., & Rockwood, A. (2001). Generalized homogeneous coordinates for computational geometry. In G. Sommer (Ed.), *Geometric computing with Clifford algebra* (pp. 27–59). New York: Springer-Verlag.
2. Bayro-Corrochano, E. (1995). Robot perception and action using conformal geometry. In E. Bayro-Corrochano (Ed.), *Handbook of geometric computing. Applications in pattern recognition, computer vision, neurocomputing and robotics* (Chap. 13, pp. 405–458). Heidelberg: Springer-Verlag.
3. Hestenes, D. (2001). Old wine in new bottles: A new algebraic framework for computational geometry. In E. Bayro-Corrochano & G. Sobczyk (Eds.), *Geometric algebra applications with applications in science and engineering* (pp. 3–17). Boston: Birkhäuser.
4. Hestenes, D. (2009). New tools for computational geometry and rejuvenation of screw theory. In E. Bayro-Corrochano & G. Sheuermann (Eds.), *Geometric algebra computing for engineering and computer science* (pp. 3–33). London: Springer.
5. Dress, A., & Havel, T. (1993). Distance geometry and geometric algebra. *Foundations of Physics*, *23*(10), 1357–1374.
6. Perwass, C. B. U. (2009). *Geometric algebra with applications in engineering*. Berline, Heidelberg: Springer Verlag.

7. Perwass, C. B. U. (2000). *Applications of geometric algebra in computer vision*. Ph.D. Thesis, University of Cambridge.
8. Needham, T. (1997). *Visual complex analysis*. New York: Oxford University Press (Reprinted 2003)
9. Rosenhan, B., Perwass, C., & Sommer, G. (2005). Pose estimation of 3D free-form curves. *Journal of Computer Vision*, *62*(3), 267–289.
10. Lounesto, P. (1987). CLICAL software packet and user manual. Helsinki University of Technology of Mathematics, Research report A248.
11. Ablamowicks, R. eCLIFFORD Software packet using Maple for Clifford algebra. Computations. http://math.tntech.edu/rafal.
12. Perwass, C. B. U. (2006). CLUCal. http://www.clucal.info/.

Chapter 7
The Geometric Algebras $G^+_{6,0,2}$, $G_{6,3}$, $G^+_{9,3}$, $G^+_{6,0,6}$

7.1 Introduction

The geometric algebra of a 3D Euclidean space $G_{3,0,0}$ has a point basis and the motor algebra $G^+_{3,0,1}$ a line basis. In the latter, the lines expressed are expressed in terms of Plücker coordinates and the points and planes in terms of bivectors. The reader can find a comparison of representations of points, lines, and planes using vector calculus, $G_{3,0,0}$ and $G^+_{3,0,1}$ in Chap. 5. Extending the degrees of freedom of the mathematical system, in the conformal geometric algebra $G_{4,1}$, using the horosphere framework points, one can model lines, planes, circles, and spheres and also certain Lie groups as a versors.

For the applications, we are interested to model not only diverse geometric entities but also their associated Lie groups. The $SE(3)$ group is modeled in $G_{3,0,0}$ in an additive manner using a rotor $\boldsymbol{R}$ and for the translation a vector $\boldsymbol{t}$. In the motor algebra $G^+_{3,0,1}$ [1, 2] with $\begin{bmatrix} 4 \\ 2 \end{bmatrix} = 6$ bivector basis, we represent $SE(3)$ using the versor called motor $\boldsymbol{M} = \boldsymbol{T}\boldsymbol{R}$ which carries out the group action, however, in a multiplicative manner. As explained in Chap. 6, the conformal geometric algebra $G_{4,1}$ of the 3D space [2] is equipped with $\begin{bmatrix} 5 \\ 2 \end{bmatrix} = 10$ bivectors which as Lie algebra generators suffice to build $SE(3)$ and extend to the conformal group involving $SE(3)$, dilations and inversions. Note that the dilations are simply isotropic transformations and do not achieve complete affine transformations. Considering other more involved Lie groups, in Sect. 3.7, in the framework $G_{3,3}$, we formulated for the 3D projective geometry its Lie group $SL(4)$ and its Lie algebra $sl(4)$ which indeed are useful for complete affine transformations as well.

Along these lines of thought, it would be beneficial to extend the degrees of freedom of certain geometric algebras so that we can model the kinematics of a variety of geometric entities together with other Lie groups. For that, we need to extend the degrees of freedom of the geometric algebra in question so that we have enough null vectors for the modeling of the geometric entities and the bivectors for generating

© Springer Nature Switzerland AG 2020

E. Bayro-Corrochano, *Geometric Algebra Applications Vol. II*,
https://doi.org/10.1007/978-3-030-34978-3_7

the required Lie algebras. In the horizon of our work in 3D, we require points, lines, circles, planes, and spheres together with the affine and projective groups, for example, to transform spheres in ellipsoids varying its orientation and position. We have formulated for the 3D projective geometry in $G_{3,3}$ the Lie group $SL(4)$ and its Lie algebra $sl(4)$, so we should now extend the number of the null vectors of the geometric algebras like $G_{3,0,0}$ and $G_{4,1}$ so that we have enough resources for this purpose, so the more natural geometric algebras appear to be $G_{6,3}$, introduced by Zamora [], which extends to three the number of null vectors and of the 3D Euclidean $G_{3,0,0}$, as the $G_{4,1}$ has just two one null vectors: one to represent the origin and another for one point at infinity; in $G_{6,3}$, we can model three orthogonal points at the origin and three at infinity which help to model projective and affine transformations. Another interesting geometric algebra is the $G^+_{6,0,2}$ which in fact is a result of the doubling procedure applied to motor algebra $G^+_{3,0,1}$, and it was proposed by Selig and Bayro-Corrochano [3] to model the kinematics and dynamics of points, lines, and planes. Another interesting geometric algebra is $G_{8,2}$, introduced by Easter and Hitzer [4], which again is the result obtained after the double procedure applied to $G_{4,1}$. In the next subsections, we will briefly describe $G^+_{6,0,2}$, $G_{6,3}$, and $G_{8,2}$ highlighting its attributes which help to overcome the limitations of the motor algebra $G^+_{3,0,2}$ and the conformal geometric algebra $G_{4,1}$.

7.2 The Double Motor Algebra $G^+_{6,0,2}$

In Sect. 4.5.1, we discusses the motor algebra $G^+_{3,0,1}$ a suitable line geometry to model the kinematics of points, lines, and planes. Selig and Bayro-Corrochano [3] recast the dynamics of rigid bodies using the double motor algebra $G^+_{6,0,2}$. The authors show how the velocities, momenta, and inertias can be represented in terms of elements of the algebra

$$G^+_{6,0,2} = \mathtt{gen}\{1, e_i e_j, e_i e, f_i f_j, f_i f, I\}, \tag{7.1}$$

with an orthogonal set of generators e_i and f_i where $i = j = 1, 2, 3$ and two null vectors e, f; 28 bivectors and the pseudoscalars $I_e = e_1 e_2 e_3 e$, $I_f = f_1 f_2 f_3 f$ and $I = e_1 e_2 e_3 e f_1 f_2 f_3 f = I_e I_f$. Note that one can transform a set of six bivectors $e_i e_j$, $e_i e$ to its orthogonal set $f_i f_j$, $f_i f$ using the pseudoscalar I, namely,

$$f_i f_j = I e_i e_j, \quad f_i f = I e_i e, \tag{7.2}$$

for $i, j = 1, 2, 3$. The Lie algebra of the Lie group of rigid motion $SE(3)$ can be represented as a six-dimensional velocity vectors called twists or sometimes for simplicity screws

$$\boldsymbol{s} = w_x e_2 e_3 + w_y e_3 e_1 + w_z e_1 e_2 + v_x e_1 e + v_y e_2 e + v_z e_3 e = \boldsymbol{w} + I_e \boldsymbol{v}. \tag{7.3}$$

The co-screw expanded with the dual bivector basis reads

$$\boldsymbol{s}_f = w_x f_2 f_3 + w_y f_3 f_1 + w_z f_1 f_2 + v_x f_1 f + v_y f_2 f + v_z f_3 f = \boldsymbol{w} + I_f \boldsymbol{v}. \quad (7.4)$$

The Lie bracket of a pair of screws $\boldsymbol{s}_1$ and $\boldsymbol{s}_2$ is computed using their commutator

$$[\boldsymbol{s}_1, \boldsymbol{s}_2] = \frac{1}{2}(\boldsymbol{s}_1 \boldsymbol{s}_2 - \boldsymbol{s}_2 \boldsymbol{s}_1). \quad (7.5)$$

Instantaneously, the body is rotating about an a screw axis $\boldsymbol{L} = \boldsymbol{n} + I_e \boldsymbol{m}$ with an angular velocity $\boldsymbol{w}$ and simultaneously translating along the same screw axis with linear velocity $\boldsymbol{v}$. Similarly, momenta called co-screws can be written using the dual bivectors as follows

$$\mathcal{P} = p_x f_2 f_3 + p_y f_3 f_1 + p_z f_1 f_2 + q_x f_1 f + q_y f_2 f + q_z f_3 f = \boldsymbol{p} + I_f \boldsymbol{q}. \quad (7.6)$$

The wrenches or co-screws are the dual of the twists or screws. These co-screws can be also used to represent wrenches which consist of torque and force bivectors

$$\boldsymbol{w} = \tau_x e_2 e_3 + \tau_y e_3 e_1 + \tau_z e_1 e_2 + F_x f_3 f_1 + F_y f_3 f_1 + F_z f_1 f_2 = \boldsymbol{\tau} + I_e \boldsymbol{F}. \quad (7.7)$$

For the transformations of screws, we utilize the homogeneous Eq. (5.34) for the case of the $SE(3)$

$$\mathtt{T} = \begin{bmatrix} \mathtt{R} & \mathtt{O} \\ [\mathtt{t}]_\times \mathtt{R} & \mathtt{R} \end{bmatrix}. \quad (7.8)$$

This transformation of lines can be formulated using motors as explained in Sect. 4.5.2

$$\begin{aligned} \boldsymbol{M} &= \boldsymbol{T}\boldsymbol{R} = e^{\frac{\theta}{2}\boldsymbol{n} + e\frac{t}{2}}, \\ \boldsymbol{M}\widetilde{\boldsymbol{M}} &= 1. \end{aligned} \quad (7.9)$$

Given a Lie algebra element or bivector $\boldsymbol{B}$, the exponential map from the Lie algebra to the Lie group is given by

$$\boldsymbol{M} = e^{\frac{\boldsymbol{B}}{2}} = 1 + (\boldsymbol{B}/2) + \frac{1}{2}(\boldsymbol{B}/2)^2 + \cdots. \quad (7.10)$$

Note that the factor $\frac{1}{2}$ is due to that we are working in the double covering group $SE(3)$. We can express a motor $\boldsymbol{M}$ using the basis of $G^+_{3,0,1}$ and $\boldsymbol{M}_f$ using the other set of bivectors of the extended $G^+_{3,0,2}$ as follows:

$$\begin{aligned} \boldsymbol{M} &= m_0 + m_1 e_2 e_3 + m_2 e_3 e_1 + m_3 e_1 e_2 + m_4 e_1 e + m_5 e_2 e + m_6 e_3 e + m_8 e, \\ \boldsymbol{M}_f &= m_0 + m_1 f_2 f_3 + m_2 f_3 f_1 + m_3 f_1 f_2 + m_4 f_1 f + m_5 f_2 f + m_6 f_3 f + m_8 f. \end{aligned}$$

According to Eq. (19.8), the dynamic motor equation is

$$\dot{\boldsymbol{M}} = -\frac{1}{2}\boldsymbol{s}\boldsymbol{M}. \tag{7.11}$$

Assuming that the screw motion of the body is constant through time, the dynamic motor Eq. (19.8) can be integrated to give

$$\boldsymbol{M}(t) = e^{-\frac{\boldsymbol{s}}{2}}\boldsymbol{M}(0) = e^{-\frac{(\boldsymbol{w}+I\boldsymbol{v})}{2}}\boldsymbol{M}(0), \tag{7.12}$$

where $\boldsymbol{s}$ is the twist. This last equation represents a motor that rotates with a constant-frequency rotation in the right-hand sense and has a constant linear velocity as well. The motor expanded with the co-screw is given by

$$\boldsymbol{M}_f(t) = e^{-\frac{\boldsymbol{s}_f}{2}}\boldsymbol{M}_f(0) = e^{-\frac{(\boldsymbol{w}+I_f\boldsymbol{v})}{2}}\boldsymbol{M}_f(0). \tag{7.13}$$

The Lie group action $SE(3)$ on lines is given by

$$\boldsymbol{L}' = \boldsymbol{M}\boldsymbol{L}\widetilde{\boldsymbol{M}} = \boldsymbol{T}\boldsymbol{R}\boldsymbol{L}\widetilde{\boldsymbol{R}}\widetilde{\boldsymbol{T}}, \tag{7.14}$$

on screws is

$$\boldsymbol{s}' = \boldsymbol{M}\boldsymbol{s}\widetilde{M}, \tag{7.15}$$

and on co-screws is

$$\boldsymbol{s}'_f = \boldsymbol{M}_f\boldsymbol{s}_f\widetilde{M}_f. \tag{7.16}$$

The group action $SE(3)$ on momenta is

$$\mathcal{P}' = \boldsymbol{M}_f\mathcal{P}\widetilde{M}_f. \tag{7.17}$$

The kinetic energy of the rigid body is an scalar function computed via the inner product of its momenta co-screw and its velocity screw

$$\begin{aligned} E_K &= \frac{1}{2}\mathcal{P}\cdot\boldsymbol{s}, \\ &= \frac{1}{2}(\boldsymbol{w}\cdot\boldsymbol{q} + \boldsymbol{v}\cdot\boldsymbol{p}) = \frac{1}{2}(w_x q_x + w_l dy q_y + w_z q_z + v_x p_x + v_y p_y + v_z p_z). \end{aligned} \tag{7.18}$$

The inertia matrix of a rigid body is given by a 6×6 matrix

$$\mathtt{N} = \begin{bmatrix} \mathtt{I} & m[\mathtt{c}]_\times \\ m[\mathtt{c}]_\times^{\mathtt{T}} & m\mathtt{I}_3 \end{bmatrix}, \tag{7.19}$$

where $\mathtt{I}$ is the 3×3 inertia matrix and $\mathtt{I}_3$ is the identity matrix, m is the mass of the body, and $\mathtt{c}$ is the position vector of the center of mass expressed as $[\mathtt{c}]_\times$ an antisymmetric matrix to compute its cross product. Since $G_{6,0,2}$ has not enough null vectors as in $G_{3,3}$ or $G_{6,3}$, we cannot express $\mathtt{N}$ as an affine transformation using bivectors as it is shown in Sect. 3.7. The inertia maps velocities to momenta; we will then formulate this mapping in terms of 6D vectors

$$\begin{bmatrix} q_x \\ q_y \\ q_z \\ p_x \\ p_y \\ p_z \end{bmatrix} = \mathtt{N} \begin{bmatrix} w_x \\ w_y \\ w_z \\ v_x \\ v_y \\ v_z \end{bmatrix} = \begin{bmatrix} \mathtt{I}\mathbf{w} + \mathtt{m}\mathbf{c} \times \mathbf{v} \\ m(\mathbf{c} \times \mathbf{w} + \mathbf{v}) \end{bmatrix}, \tag{7.20}$$

which in terms of bivectors can be written as

$$\mathtt{N} : \boldsymbol{s} \rightarrow \mathcal{P} = \boldsymbol{q} + I_f \boldsymbol{p} = (\mathtt{I}\boldsymbol{w} + \mathtt{m}(\boldsymbol{c} \times \boldsymbol{v})) + \mathtt{I}_\mathtt{e}\mathtt{m}(\boldsymbol{c} \times \boldsymbol{w} + \boldsymbol{v}). \tag{7.21}$$

The transformation properties of the inertia matrix can be inferred based on the fact that the kinetic energy must be invariant, namely,

$$\mathtt{N} \rightarrow \mathtt{T}^{-\mathtt{T}}\mathtt{N}\mathtt{T}^{-1}. \tag{7.22}$$

7.2.1 The Shuffle Product

The dual of the join operator for expansion is the meet product for intersection. In non-degenerated geometric algebras, the meet is computed using the pseudoscalar I, namely,

$$\boldsymbol{A} \vee \boldsymbol{B} = (\boldsymbol{A}^* \wedge \boldsymbol{B}^*)I = (\boldsymbol{A}I^{-1} \wedge \boldsymbol{B}I^{-1})I. \tag{7.23}$$

Since $G^{+}_{6,0,2}$ is degenerated, the I^{-1} doesn't exist, and thus we need to resort to the shuffle product of homogeneous k-vectors which yields as result the common vectors existing in both k-vectors. According to [5], given $\boldsymbol{A} = a_1 \wedge a_2 \wedge \cdots \wedge a_j$ and $\boldsymbol{B} = b_1 \wedge b_2 \wedge \cdots \wedge b_k$, their shuffle product is computed as follows:

$$\boldsymbol{A} \vee \boldsymbol{B} = \sum_{\sigma} \text{sign}(\sigma) \det(a_{\sigma(1)}, \ldots, a_{\sigma(n-k)}, b_1, \ldots, b_k) a_{\sigma(n-k+1)} \wedge \cdots \wedge a_{\sigma(j)}. \tag{7.24}$$

The sum is taken over all permutations σ of $1, 2, \ldots, j$ such that $\sigma(1) < \sigma(2) < \cdots < \sigma(n-k)$ and $\sigma(n-k+1) < \sigma(n-k+2) < \cdots < \sigma(j)$. Each a_i can be written as a sum of basis elements

$$a_i = a_{i1}e_1 + a_{i2}e_2 + \cdots + a_{in}e_n. \tag{7.25}$$

The determinant in Eq. (7.24) is the determinant of the matrix whose columns are the coefficients $a_{\sigma(1)i}, a_{\sigma(2)i}, \cdots + b_{\sigma(k)i}$. When $j + k < n$ the shuffle product is zero. As an example considered in $G^+_{6,0,2}$, the shuffle product between $f_1 f_2 f_3 f$ and $f_3 f_1 e_1 e_2 e_3 e$

$$\begin{aligned} f_1 f_2 f_3 f \vee f_3 f_1 e_1 e_2 e_3 e &= \det(f_1 f_2 f_3 f_1 e_1 e_2 e_3 e) f_3 f - \\ &- \det(f_2 f f_3 f_1 e_1 e_2 e_3 e) f_1 f_3 + \cdots + \det(f_3 f f_3 f_1 e_1 e_2 e_3 e) f_1 f_2, \\ &= f_3 f_1. \end{aligned} \tag{7.26}$$

Note that the determinant values are either zero or 1; only the determinant of the middle term is 1.

7.2.2 Equations of Motion

In order to formulate the equation of motion of a rigid body, we need the co-adjoint action of a screw on a co-screw; in this way, one can compute the Coriolis terms. The operation is dual to the adjoint action of the screws acting on themselves involving a Lie bracket, namely,

$$\boldsymbol{s}_1^T \{\boldsymbol{s}_2, \mathcal{P}\} = [\boldsymbol{s}_1, \boldsymbol{s}_2]^T \mathcal{P}, \tag{7.27}$$

which written as vectors read

$$\{\boldsymbol{s}, \mathcal{P}\} = \left\{ \begin{pmatrix} \boldsymbol{w} \\ \boldsymbol{v} \end{pmatrix}, \begin{pmatrix} \boldsymbol{q} \\ \boldsymbol{p} \end{pmatrix} \right\} = \begin{pmatrix} \boldsymbol{w} \times \boldsymbol{q} + \boldsymbol{v} \times \boldsymbol{p} \\ \boldsymbol{w} \times \boldsymbol{p} \end{pmatrix}. \tag{7.28}$$

Thus, the dynamic equation can be written as follows

$$\mathtt{N}\frac{d}{dt}\boldsymbol{s} + \{\boldsymbol{s}, \mathtt{N}\,\boldsymbol{s}\} = \mathtt{N}\frac{d}{dt}\boldsymbol{s} + \{\boldsymbol{s}, \mathcal{P}\} = \boldsymbol{\tau}, \tag{7.29}$$

where $\mathtt{N}$ is a 6×6 tensor applied to $\boldsymbol{s}$ and $\boldsymbol{\tau}$ is the external wrench (force/torque co-screw) acting on the body.

Next, we will formulate Eq. (7.29) entirely in the geometric algebra $G^+_{6,0,2}$. The evaluation map of a co-screw on a screw can be formulated using the following invariant element:

$$Q = f_2 f_3 e_1 e + f_3 f_1 e_2 e + f_1 f_2 e_3 e + f_1 f e_2 e_3 + f_2 f e_3 e_1 + f_3 f e_1 e_2.$$

The invariance of Q is with respect to the following group action $Q = (\boldsymbol{M}\boldsymbol{M}_f)$ $Q(\widetilde{\boldsymbol{M}}_f \widetilde{\boldsymbol{M}})$. The evaluation map can be formulated using the shuffle operator $\vee$ as follows:

$$\mathcal{P}(\boldsymbol{s}) = (\mathcal{P} \wedge Q) \vee \boldsymbol{s} = \mathcal{P} \vee (Q \wedge \boldsymbol{s}). \tag{7.30}$$

Since

$$\begin{aligned} Q \wedge \boldsymbol{s} &= (w_x f_2 f_3 + w_y f_3 f_1 + w_z f_1 f_2 + v_x f_1 f + v_y f_2 f + v_z f_3 f) e_1 e_2 e_3 e, \\ &= (w_x f_2 f_3 + w_y f_3 f_1 + w_z f_1 f_2 + v_x f_1 f + v_y f_2 f + v_z f_3 f) I_e. \end{aligned} \tag{7.31}$$

Then evaluating the shuffle operation $\vee$, we get

$$\mathcal{P}(\boldsymbol{s}) = \mathcal{P} \vee (Q \wedge \boldsymbol{s}) = w_x q_x + w_y q_y + w_z q_z + v_x p_x + v_y p_y + v_z p_z. \tag{7.32}$$

The inertia matrix can be transformed into a diagonal one via a suitable rigid body motion. The resulting diagonal matrix rewritten in terms of bivectors reads

$$\boldsymbol{N} = d_x f_1 f e_1 e + d_y f_2 f e_2 e + d_z f_3 f_2 e_3 e + m f_2 f_3 e_2 e_3 + m f_3 f_1 e_3 e_1 + m f_1 f_2 e_1 e_2,$$

where m stands for the body mass and the d_i is the mass times the square of the radius of gyration about the principle axis i.

In order to construct the map from velocities to momenta, we use another invariant, the pseudoscalar $I_f = f_1 f_2 f_3 f$. This element commutes with any $\boldsymbol{M}$, i.e., $I_f = \boldsymbol{M} I_f \widetilde{\boldsymbol{M}}$. The invariance of I_f is with respect to the group action $I_f = (\boldsymbol{M}\boldsymbol{M}_f) I_f (\widetilde{\boldsymbol{M}}_f \widetilde{\boldsymbol{M}})$ as well.

The evaluation map from velocities to momenta is given by

$$\mathcal{P}(\boldsymbol{s}) = (I_f \wedge \boldsymbol{N}) \vee s = I_f \vee (\boldsymbol{N} \wedge \boldsymbol{s}). \tag{7.33}$$

Similar as in computing Eq. (7.30) using the bivector inertia one obtains

$$\mathcal{P} = m v_x f_2 f_3 + m v_y f_3 f_1 + m v_z f_1 f_2 + d_x w_x f_1 f + d_y w_y f_2 f + d_z w_z f_3 f. \tag{7.34}$$

Next, let us study the action of $SE(3)$ on the inertia. First, we transform the moment co-screw using $\boldsymbol{M}\boldsymbol{M}_f$

$$\begin{aligned} (\boldsymbol{M}\boldsymbol{M}_f)\mathcal{P}(\widetilde{\boldsymbol{M}}_f \widetilde{\boldsymbol{M}}) &= (\boldsymbol{M}\boldsymbol{M}_f)(I_f \vee (\boldsymbol{N} \wedge s)(\widetilde{\boldsymbol{M}}_f \widetilde{\boldsymbol{M}})), \\ &= (\boldsymbol{M}\boldsymbol{M}_f) I_f (\widetilde{\boldsymbol{M}}_f \widetilde{\boldsymbol{M}}) \vee (\boldsymbol{M}\boldsymbol{M}_f) \boldsymbol{N} (\widetilde{\boldsymbol{M}}_f \widetilde{\boldsymbol{M}}) \wedge (\boldsymbol{M}\boldsymbol{M}_f) \boldsymbol{s} (\widetilde{\boldsymbol{M}}_f \widetilde{\boldsymbol{M}}). \end{aligned} \tag{7.35}$$

Since I_f is invariant, $\mathcal{P}$ commutes with $\boldsymbol{M}$ and $\boldsymbol{s}$ commutes with $\boldsymbol{M}_f$, we obtain

$$\boldsymbol{M}_f \mathcal{P} \widetilde{\boldsymbol{M}}_f \widetilde{\boldsymbol{M}} = I_f \vee (\boldsymbol{M}\boldsymbol{M}_f \boldsymbol{N} \widetilde{\boldsymbol{M}}_f \widetilde{\boldsymbol{M}} \wedge \boldsymbol{M} \boldsymbol{s} \widetilde{\boldsymbol{M}}). \tag{7.36}$$

Thus, we can infer that the inertia, the screw, and co-screw must transform according to

$$\begin{aligned} \boldsymbol{N} &\rightarrow (\boldsymbol{M}\boldsymbol{M}_f)\boldsymbol{N}(\widetilde{\boldsymbol{M}}_f\widetilde{\boldsymbol{M}}), \\ \boldsymbol{s} &\rightarrow (\boldsymbol{M}\boldsymbol{M}_f)\boldsymbol{s}(\widetilde{\boldsymbol{M}}_f\widetilde{\boldsymbol{M}}), \\ \mathcal{P} &\rightarrow (\boldsymbol{M}\boldsymbol{M}_f)\mathcal{P}(\widetilde{\boldsymbol{M}}_f\widetilde{\boldsymbol{M}}). \end{aligned} \tag{7.37}$$

Let us consider how the diagonal inertia is transformed under a motor for translation or translator in the y-direction $\boldsymbol{M} = 1 + \frac{1}{2}(t_y e_2 e)$; thus

$$\boldsymbol{M}\boldsymbol{M}_f = \left(1 + \frac{1}{2}t_y e_2 e\right)\left(1 + \frac{1}{2}t_y f_2 f\right) = 1 + \frac{1}{2}t_y f_2 f + \frac{1}{2}t_y e_2 e + \frac{1}{4}t_y^2 f_2 f e_2 e.$$

Computing the action on the diagonal inertia, we get

$$\begin{aligned} (\boldsymbol{M}\boldsymbol{M}_f)\boldsymbol{N}(\widetilde{\boldsymbol{M}}_f\widetilde{\boldsymbol{M}}) = {} & (d_x + mt_y^2) f_1 f e_1 e + d_y f_2 f e_2 e + (d_z + mt_y^2) f_3 f e_3 e \\ & - mt_y f_3 f e_2 e_3 + mt_y f_1 f e_1 e_2 + mt_y f_1 f_2 e_1 e - mt_y f_2 f_3 e_3 e \\ & + mf_2 f_3 e_2 e_3 + mf_3 f_1 e_3 e_1 + mf_1 f_2 e_1 e_2. \end{aligned} \tag{7.38}$$

We can compare this result expressed in terms of bivectors with the corresponding 6×6 inertia matrix

$$\begin{bmatrix} d_x + mt_y^2 & 0 & 0 & 0 & 0 & mt_y \\ 0 & d_y & 0 & 0 & 0 & 0 \\ 0 & 0 & d_z + mt_y^2 & -mt_y & 0 & 0 \\ 0 & 0 & -mt_y & m & 0 & 0 \\ 0 & 0 & 0 & 0 & m & 0 \\ mt_y & 0 & 0 & 0 & 0 & m \end{bmatrix}. \tag{7.39}$$

The kinetic energy of the rigid body can be written as

$$E_k = \frac{1}{2}\mathcal{P} \vee (Q \wedge \boldsymbol{s}) = \frac{1}{2} I_f \vee (Q \wedge \boldsymbol{s}) \vee (\boldsymbol{N} \wedge \boldsymbol{s}). \tag{7.40}$$

Note that the bivector

$$I_f \vee (Q \wedge \boldsymbol{s}) = w_x f_2 f_3 + w_y f_3 f_1 + w_z f_1 f_2 + v_x f_1 f + v_y f_2 f + v_z f_3 f \tag{7.41}$$

is the same as the screw expanded with the bivectors basis $f_j f$ called $\boldsymbol{s}_f$; thus, we can rewrite the previous equation in a more compact manner as follows:

$$E_k = \frac{1}{2}\boldsymbol{s}_f \vee (\boldsymbol{N} \wedge \boldsymbol{s}). \tag{7.42}$$

According to Eq. (7.13), the group action on the co-screw momenta as function of time can be written as

$$\mathcal{P}'(t) = \boldsymbol{M}_f(t)\mathcal{P}\widetilde{\boldsymbol{M}}_f(t) = e^{t\boldsymbol{s}_f}\mathcal{P}e^{-t\boldsymbol{s}_f}. \tag{7.43}$$

As usual, to get the Lie algebra bracket, if we differentiate the last equation with respect to t and then set $t = 0$, we obtain the co-adjoint action of the screw $\boldsymbol{s}$ on the co-screw $\mathcal{P}$,

$$\begin{aligned} \{\boldsymbol{s}_f, \mathcal{P}\} &= \frac{1}{2}(\boldsymbol{s}_f\mathcal{P} - \mathcal{P}\boldsymbol{s}_f), \\ \{\boldsymbol{s}_f, I_f \vee (\boldsymbol{N} \wedge \boldsymbol{s})\} &= \frac{1}{2}(\boldsymbol{s}_f(I_f \vee (\boldsymbol{N} \wedge \boldsymbol{s})) - (I_f \vee (\boldsymbol{N} \wedge \boldsymbol{s}))\boldsymbol{s}_f). \end{aligned} \tag{7.44}$$

According to Newton's second law, the force applied to the body is equal to the rate of change of momentum

$$\frac{d}{dt}\mathcal{P} = \boldsymbol{\tau}, \tag{7.45}$$

where $\boldsymbol{\tau}$ is the applied wrench. Next, for simplicity we do not include explicitly the variable t in Eq. 7.38, and it is rewritten in terms of the bivector exponents

$$\boldsymbol{N}' = (\boldsymbol{M}\boldsymbol{M}_f)\boldsymbol{N}(\widetilde{\boldsymbol{M}}_f\widetilde{\boldsymbol{M}}) = e^{\frac{t}{2}(\boldsymbol{s}+\boldsymbol{s}_f)}\boldsymbol{N}e^{-\frac{t}{2}(\boldsymbol{s}_f+\boldsymbol{s}_f)}. \tag{7.46}$$

Then differentiating this equation with respect to t and setting $t = 0$, we get

$$\frac{d}{dt}\boldsymbol{N} = \frac{1}{2}(\boldsymbol{s}\boldsymbol{N} - \boldsymbol{N}\boldsymbol{s}) + \frac{1}{2}(\boldsymbol{s}_f\boldsymbol{N} - \boldsymbol{N}\boldsymbol{s}_f). \tag{7.47}$$

According to Eq. (7.33), the momentum is given by

$$\mathcal{P}(\boldsymbol{s}) = I_f \vee (\boldsymbol{N} \wedge \boldsymbol{s}). \tag{7.48}$$

Since $\boldsymbol{I}_f$ is an invariant, its derivative is zero and $(\boldsymbol{s}\boldsymbol{N} - \boldsymbol{N}\boldsymbol{s}) \wedge \boldsymbol{s} = 0$; thus, we can simplify the derivative of $\mathcal{P}$ equation (7.45) as follows:

$$\frac{d}{dt}\mathcal{P} = I_f \vee (\boldsymbol{N} \wedge \dot{\boldsymbol{s}}) + \frac{1}{2}I_f((\boldsymbol{s}_f\boldsymbol{N} - \boldsymbol{N}\boldsymbol{s}_f) \wedge \boldsymbol{s}). \tag{7.49}$$

Note that the second term on the right-hand side of previous equation is similar to Eq. (7.44).

Furthermore, we can improve the motion equation using the invariant $I_e = e_1e_2e_3e$, since $I_fI_e \vee \boldsymbol{p} = I \vee \boldsymbol{p} = \boldsymbol{p}$ for any element $\boldsymbol{p} \in G_{6,0,2}$; multiplying previous equation by I_e, we obtain

$$\boldsymbol{N} \wedge \dot{\boldsymbol{s}} + \frac{1}{2}((\boldsymbol{s}_f \boldsymbol{N} - \boldsymbol{N}\boldsymbol{s}_f) \wedge \boldsymbol{s}) = \boldsymbol{\tau} I_e. \tag{7.50}$$

7.3 The Geometric Algebra $G_{6,3}$

The $G_{6,3}$ was introduced by Zamora-Esquivel [6]; in his work, he describes the modeling of the geometric entities and certain Lie groups using versors. In this subsection, we will study in detail the $G_{6,3}$. Note that $G_{6,3}$ includes the 3D geometric algebra, G_3, and the mother algebra of the 3D Euclidean space $G_{3,3}$ useful for projective and affine geometry. Note that $G_{3,0}$ can be extended to the conformal GA $G_{4,1}$ to have a couple of null vectors e_0 and e_∞ and further to $G_{6,3}$ to have three couples of null vectors $\{e_{01}, e_{\infty 1}\}$, $\{e_{02}, e_{\infty 2}\}$, and $\{e_{03}, e_{\infty 3}\}$. The definitions given below are based in concepts of the geometric algebra of the reciprocal null cones explained in Sect. 3.5 and in conformal geometric algebra of Chap. 6.

The unit vector basis of $G_{6,3}$ is $\{e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8, e_9\}$ with the following properties:

$$\begin{aligned} e_i^2 &= 1, \quad i = 1, 2, 3; \\ e_j^2 &= 1, \quad j = 4, 5, 6; \\ e_k^2 &= -1, \quad k = 7, 8, 9; \\ e_i \cdot e_j &= e_i \cdot e_k = e_j \cdot e_k = 0. \end{aligned} \tag{7.51}$$

Note that this basis is not written in bold. We can form a set of null basis consisting of three couples of null vectors $\{e_{0i}, e_{\infty i}\}$ for $i = 1, 2, 3$ as follows:

$$\begin{aligned} e_{01} &= \frac{(e_7 - e_4)}{2}, \quad e_{\infty 1} = e_7 + e_4, \\ e_{02} &= \frac{(e_8 - e_5)}{2}, \quad e_{\infty 2} = e_8 + e_5, \\ e_{03} &= \frac{(e_9 - e_6)}{2}, \quad e_{\infty 3} = e_9 + e_6, \end{aligned}$$

with the properties

$$e_{0i}^2 = e_{\infty i}^2 = 0, \quad e_{\infty i} \cdot e_{0i} = -1. \tag{7.52}$$

The 3D Euclidean pseudoscalar I_e and the pseudoscalar I of $G_{6,3}$ are defined by

$$I_e = e_1 e_2 e_3 \qquad I = e_1 e_2 e_3 e_4 e_5 e_6 e_7 e_8 e_9. \tag{7.53}$$

The unit pseudoscalars $\boldsymbol{E}_1$, $\boldsymbol{E}_2$, and $\boldsymbol{E}_3$ represent three Minkowski planes and their wedge a 6-vector unit pseudoscalar $\boldsymbol{E}_{123} \in \mathbb{R}^{3,3}$ as follows:

$$\begin{aligned}
\boldsymbol{E}_1 &= e_{\infty 1} \wedge e_{01} = e_4 \wedge e_7 = e_4 e_7, \\
\boldsymbol{E}_2 &= e_{\infty 2} \wedge e_{02} = e_5 \wedge e_8 = e_5 e_8, \\
\boldsymbol{E}_3 &= e_{\infty 3} \wedge e_{03} = e_6 \wedge e_9 = e_6 e_9, \\
\boldsymbol{E}_{123} &= \boldsymbol{E}_1 \wedge \boldsymbol{E}_2 \wedge \boldsymbol{E}_3 = \boldsymbol{E}_1 \boldsymbol{E}_2 \boldsymbol{E}_3
\end{aligned} \tag{7.54}$$

having the properties

$$\begin{aligned}
\boldsymbol{E}_1^2 = \boldsymbol{E}_2^2 = \boldsymbol{E}_3^2 = 1, \\
\widetilde{\boldsymbol{E}}_1 = -\boldsymbol{E}_1, \;\; \widetilde{\boldsymbol{E}}_2 = -\boldsymbol{E}_2, \;\; \widetilde{\boldsymbol{E}}_3 = -\boldsymbol{E}_3, \\
\boldsymbol{E}_{123}^2 = 1, \;\; \widetilde{\boldsymbol{E}}_{123} = -\boldsymbol{E}_{123}, \\
\boldsymbol{E}_1 e_{\infty 1} = -e_{\infty 1} \boldsymbol{E}_1 = -e_{\infty_1}, \;\; \boldsymbol{E}_1 e_{01} = -e_{01} \boldsymbol{E}_1 = e_{01} \quad \text{(absorption)}, \\
1 - \boldsymbol{E}_1 = -e_{\infty 1} e_{01}, \;\; 1 + \boldsymbol{E}_1 = -e_{01} e_{\infty 1}, \\
\boldsymbol{E}_1 e_{\infty 1} = -e_{\infty 1} \boldsymbol{E}_1 = -e_{\infty_1}, \;\; \boldsymbol{E}_1 e_{01} = -e_{01} \boldsymbol{E}_2 = e_{02}, \\
1 - \boldsymbol{E}_2 = -e_{\infty 2} e_{02}, \;\; 1 + \boldsymbol{E}_2 = -e_{02} e_{\infty 2}, \\
\boldsymbol{E}_3 e_{\infty 3} = -e_{\infty 3} \boldsymbol{E}_3 = -e_{\infty 3}, \;\; \boldsymbol{E}_3 e_{03} = -e_{03} \boldsymbol{E}_3 = e_{03}, \\
1 - \boldsymbol{E}_3 = -e_{\infty 3} e_{03}, \;\; 1 + \boldsymbol{E}_3 = -e_{03} e_{\infty 3}.
\end{aligned} \tag{7.55}$$

The unit pseudoscalar of $G_{6,3}$ can be computed as $I = I_e E_{123}$. The duals of $\boldsymbol{E}_i$ and $\boldsymbol{E}_{123}$ are given by

$$\begin{aligned}
\boldsymbol{E}_i{}^* &= \boldsymbol{E}_i I^{-1} = -\boldsymbol{E}\widetilde{I}, \;\; \texttt{for i} = 1, 2, 3 \\
\boldsymbol{E}_{123}^* &= \boldsymbol{E}_{123} I^{-1} = -\boldsymbol{E}\tilde{I}.
\end{aligned} \tag{7.56}$$

Note that $\mathbb{R}^{6,3} = \mathbb{R}^3 \oplus \mathbb{R}^{3,3}$. We define other pseudoscalars which can be used to change between a IPNS representation to an OPNS one

$$I_i = I_e \wedge e_{\infty 1} \wedge e_{\infty 2} \wedge e_{\infty 3} \wedge e_0 \qquad I_o = I_e \wedge e_{01} \wedge e_{02} \wedge e_{03} \wedge e_\infty. \tag{7.57}$$

7.3.1 Additive Split of $G_{6,3}$

Similar as in conformal geometric algebra of Chap. 6, we can compute the origin and a point at infinity using the stereographic projection with respect to three projection axes formed with three origins and three points at infinity. Given an Euclidean point $\mathbf{x}_e = xe_1 + ye_2 + ze_3 \in \mathbb{R}^3$, we map into $G_{6,3}$ as follows:

$$\begin{aligned}
\boldsymbol{x}_w = \; & 2\frac{x}{x^2+1} e_1 + \frac{x^2-1}{x^2+1} e_4 + e_7 \\
& + 2\frac{y}{y^2+1} e_2 + \frac{y^2-1}{y^2+1} e_5 + e_8 + 2\frac{z}{z^2+1} e_3 + \frac{z^2-1}{z^2+1} e_6 + e_9.
\end{aligned} \tag{7.58}$$

The origin is a linear combination of e_{01}, e_{02}, and e_{03}, and it is computed as follows:

$$\begin{aligned} e_0 &= \lim_{x,y,x\to 0}(\boldsymbol{x}_w) = \frac{1}{2}(e_7 - e_4) + \frac{1}{2}(e_8 - e_5) + \frac{1}{2}(e_9 - e_6), \\ &= e_{01} + e_{02} + e_{03}. \end{aligned} \tag{7.59}$$

One can interpret this result as the origin e_0 is a linear combination of three null vectors lying in general position on a plane passing through the origin e_0.

On the other hand, the point at infinity can be computed as the limit at ∞ of the stereographic projection with respect to three projection axes formed with three origins and three points at infinity, namely,

$$\begin{aligned} e_\infty &= \lim_{x,y,x\to\infty}(\boldsymbol{x}_w), \\ &= (e_7 + e_4) + (e_8 + e_5) + (e_9 + e_6) = (e_{\infty 1} + e_{\infty 2} + e_{\infty 3}), \\ &= \frac{1}{3}(e_{\infty 1} + e_{\infty 2} + e_{\infty 3}). \end{aligned} \tag{7.60}$$

Since e_∞ is homogeneous, it has been normalized by 3 just to fulfill the Lobachevsky relation $e_\infty \cdot e_0 = 1$. One can interpret this result as the point at infinity e_∞ is the linear combination of three points at infinity lying in general position on the plane at infinity π_∞.

Euclidean points $\mathbf{x}_e \in \mathbb{R}^3$ can be represented in $G_{6,3}$ as

$$\boldsymbol{x}_w = \mathbf{x}_e + \sum_{i=1,3}(\alpha_i e_{0i} + \beta_i e_{\infty i}), \tag{7.61}$$

where α_i and β_i are arbitrary scalars. Now, for an intuitive modeling of other geometric objects like ellipsoids and cylinders, the representation of $\boldsymbol{x}_w$ can be rewritten in terms of the origin e_0 and the point at infinity e_∞ as follows:

$$\boldsymbol{x}_w = xe_1 + ye_2 + ze_3 + \frac{1}{2}(x^2 e_{\infty 1} + y^2 e_{\infty 2} + z^2 e_{\infty 3}) + e_0. \tag{7.62}$$

A point $\boldsymbol{x}_w \in \mathbb{R}^{6,3}$ can be divided into its Euclidean and additive split parts by an operation called the *additive split*: $\mathbb{R}^{6,3} = \mathbb{R}^3 \oplus \mathbb{R}^{3,3}$ [7]. This additive split is defined by the projection operators $P^{||}_E$ (projection) and $P^{\perp}_E$ (rejection) as follows:

$$P^{||}_E(\boldsymbol{x}_w) = (\boldsymbol{x}_w \cdot \boldsymbol{E}_{123})\boldsymbol{E}_{123} = \sum_{i=1,3}(\alpha_i e_{0i} + \beta_i e_{\infty i}) \in \mathbb{R}^3, \tag{7.63}$$

$$P^{\perp}_E(\boldsymbol{x}_w) = (\boldsymbol{x}_w \cdot \boldsymbol{E}^*_{123})\widetilde{\boldsymbol{E}^*_{123}} = (\boldsymbol{x}_w \wedge \boldsymbol{E}_{123})\boldsymbol{E}_{123} = \mathbf{x}_e \in \mathbb{R}^{3,3}, \tag{7.64}$$

$$\boldsymbol{x}_w = P_E(\boldsymbol{x}_w) + P^{\perp}_E(\boldsymbol{x}_w). \tag{7.65}$$

7.3.2 Geometric Entities of $G_{6,3}$

Recall that in $G_{4,1}$, the basic entity is the sphere with $SE(3)$ transformation (rotations, translations) and isotropic dilations. Since in $G_{6,3}$ affine transformations are allowed, its basic entity of grade one is a quadric surface like ellipsoids, hyperboloid, spheres, and the like.

For the modeling of quadric surfaces Q, we will resort to a procedure already illustrated in Sect. 6.1.6, namely, a point $\boldsymbol{x}_w$ lying on a surface Q satisfies the following constraint

$$x_w \cdot Q = 0. \tag{7.66}$$

The classic equation of an ellipsoid is

$$\frac{(x-p)^2}{a^2} + \frac{(y-q)^2}{b^2} + \frac{(z-s)^2}{c^2} = 1, \tag{7.67}$$

which can be expanded and factorized as

$$\frac{px}{a^2}e_1 + \frac{qy}{b^2}e_2 + \frac{sz}{c^2}e_3 - \frac{1}{2}\left(\frac{p^2}{a^2} + \frac{q^2}{b^2} + \frac{s^2}{c^2} - 1\right) - \frac{1}{2}\left(\frac{x^2}{2a^2} + \frac{y^2}{2b^2} + \frac{z^2}{2c^2}\right) = 0. \tag{7.68}$$

Considering the point Eq. 7.62, we can factorize 7.68 in terms of inner product $\boldsymbol{x}_w \cdot Q = \boldsymbol{x}_w$ where the Q corresponds to the equation of an ellipsoid.

$$\begin{aligned} Q = &\frac{p}{a^2}e_1 + \frac{q}{b^2}e_2 + \frac{s}{c^2}e_3 + \frac{1}{2}\left(\frac{p^2}{a^2} + \frac{q^2}{b^2} + \frac{s^2}{c^2} - 1\right)e_\infty \\ &+ \left(\frac{1}{a^2}e_{01} + \frac{1}{b^2}e_{02} + \frac{1}{c^2}e_{03}\right). \end{aligned} \tag{7.69}$$

Note that the Q is a IPNS vector. On the other hand, the 6-vector of the dual Q^* in OPNS reads

$$Q^* = QI = \boldsymbol{x}_{w1} \wedge \boldsymbol{x}_{w2} \wedge \boldsymbol{x}_{w3} \wedge \boldsymbol{x}_{w4} \wedge \boldsymbol{x}_{w5} \wedge \boldsymbol{x}_{w6}. \tag{7.70}$$

Using Eq. 7.69 of the ellipsoid, one can straightforwardly derive the equation of the sphere, namely, using the same ratio r for the axes x, y, z and factorizing the projective scalar r^2 we get

$$
\begin{aligned}
S &= \frac{p}{r^2}e_1 + \frac{q}{r^2}e_2 + \frac{s}{r^2}e_3 + \frac{1}{2}\left(\frac{p^2}{r^2} + \frac{q^2}{r^2} + \frac{s^2}{r^2} - 1\right)e_\infty + \left(\frac{1}{r^2}e_{01} + \frac{1}{r^2}e_{02} + \frac{1}{r^2}e_{03}\right), \\
&= pe_1 + qe_2 + se_3 + \frac{1}{2}\left(p^2 + q^2 + s^2 - r^2\right)e_\infty + (e_{01} + e_{02} + e_{03}), \\
&= \boldsymbol{x}_e + \frac{1}{2}\left(\boldsymbol{x}_e^2 - r^2\right)e_\infty + e_0. \qquad (7.71)
\end{aligned}
$$

Note that the equation for the sphere is equal to the equation for the sphere as in $G_{4,1}$.

Using Eq. 7.69 of the ellipsoid, one can again straightforwardly derive the equation of the cylinder, namely, setting the ratio of axis z to infinity, we get

$$
C = \frac{p}{a^2}e_1 + \frac{q}{b^2}e_2 + \frac{1}{2}\left(\frac{p^2}{a^2} + \frac{q^2}{b^2} - 1\right)e_\infty + \left(\frac{1}{a^2}e_{01} + \frac{1}{b^2}e_{02}\right). \qquad (7.72)
$$

For the 6-vector of the dual C^* in OPNS, we set one of the points to infinity,

$$
C^* = CI = \boldsymbol{x}_{w1} \wedge \boldsymbol{x}_{w2} \wedge \boldsymbol{x}_{w3} \wedge \boldsymbol{x}_{w4} \wedge \boldsymbol{x}_{w5} \wedge \boldsymbol{e}_{\infty 3}. \qquad (7.73)
$$

The interpretation of the cylinder is that an axis z of an ellipsoid touches the plane at infinity. Figure 7.1 shows the intersection of a cylinder and an hyperboloid.

The equation for a pair of planes PP_{yz} is derived from Eq. 7.67

$$
(x - p)^2 = a^2, \qquad (7.74)
$$

which can be rewritten in terms of a inner product $\boldsymbol{x}_w \cdot PP_{yz}$. Thus, using Eq. (7.69) of the ellipsoid, one can again straightforwardly derive the equation of the pair of planes, namely, setting the ratio of axis y and z to infinity, we get

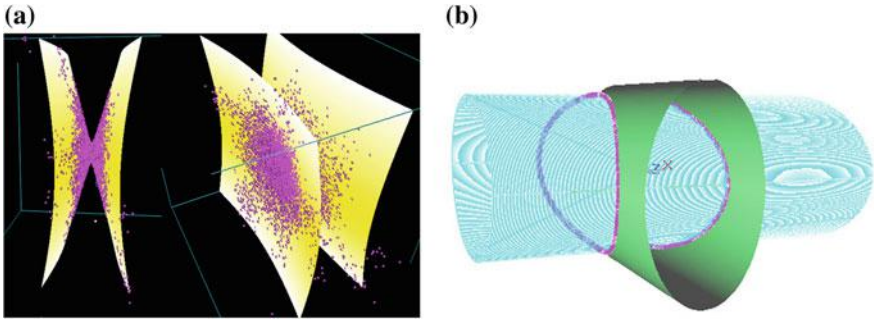


Fig. 7.1 **a** Hyperboloid in $G_{6,3}$. **b** Cylinder and hyperboloid in $G_{6,3}$. Figures provided by Julio Zamora-Esquivel

$$PP_{yz} = \frac{p}{a^2}e_1 + \frac{1}{2}\left(\frac{p^2}{a^2} - 1\right)e_\infty + \frac{1}{a^2}e_{01},$$
$$= pe_1 + \frac{1}{2}\left(p^2 - a^2\right)e_\infty + e_{01}. \tag{7.75}$$

The OPNS representation of two planes parallel to the yz-planes is formulated setting two points at infinity

$$PP^*_{yz} = \boldsymbol{x}_{w1} \wedge \boldsymbol{x}_{w2} \wedge \boldsymbol{x}_{w3} \wedge \boldsymbol{x}_{w4} \wedge \boldsymbol{e}_{\infty 2} \wedge \boldsymbol{e}_{\infty 3}. \tag{7.76}$$

The equation of the plane equals to the one in $G_{4,1}$, and it is a vector

$$\pi = n + de_\infty. \tag{7.77}$$

The OPNS representation of a plane can be computed as the wedge product of three points in general position lying on a plane and three points at infinity or the plane at infinity

$$\pi^* = \boldsymbol{x}_{w1} \wedge \boldsymbol{x}_{w2} \wedge \boldsymbol{x}_{w3} \wedge e_{\infty 1} \wedge \boldsymbol{e}_{\infty 2} \wedge \boldsymbol{e}_{\infty 3} = \boldsymbol{x}_{w1} \wedge \boldsymbol{x}_{w2} \wedge \boldsymbol{x}_{w3} \wedge \pi_\infty. \tag{7.78}$$

The equation of the line equals to the one in $G_{4,1}$, and it is bivector

$$L = nI_e + e_\infty m. \tag{7.79}$$

The OPNS representation of a line can be computed using the wedge of two points and the plane at infinity

$$L^* = \pi^* = \boldsymbol{x}_{w1} \wedge \boldsymbol{x}_{w2} \wedge e_{\infty 1} \wedge \boldsymbol{e}_{\infty 2} \wedge \boldsymbol{e}_{\infty 3} = \boldsymbol{x}_{w1} \wedge \boldsymbol{x}_{w2} \wedge \pi_\infty. \tag{7.80}$$

7.3.3 Intersection of Surfaces

An ellipse $\boldsymbol{q}$ can be regarded as the intersection of two ellipsoids $\boldsymbol{Q}_1$ and $\boldsymbol{Q}_2$. This means that for each point on the ellipse $\boldsymbol{x}_w \in \boldsymbol{q}$ they lie on both ellipsoids, that is, $\boldsymbol{x}_w \in \boldsymbol{Q}_1$ and $\boldsymbol{x}_w \in \boldsymbol{Q}_2$. Assuming that $\boldsymbol{Q}_1$ and $\boldsymbol{Q}_2$ are linearly independent, we can write for $\boldsymbol{x}_w \in \boldsymbol{q}$

$$(\boldsymbol{x}_w \cdot \boldsymbol{Q}_1)\boldsymbol{Q}_2 - (\boldsymbol{x}_w \cdot \boldsymbol{Q}_2)\boldsymbol{Q}_1 = x_w \cdot (\boldsymbol{Q}_1 \wedge \boldsymbol{Q}_2) = \boldsymbol{x}_w \cdot \boldsymbol{q} = 0. \tag{7.81}$$

This result tells us that since $\boldsymbol{x}_w$ lies on both ellipsoids,

$$\boldsymbol{q} = (\boldsymbol{Q}_1 \wedge \boldsymbol{Q}_2). \tag{7.82}$$

Thus the intersection of the ellipsoids is the ellipse $\boldsymbol{q}$.

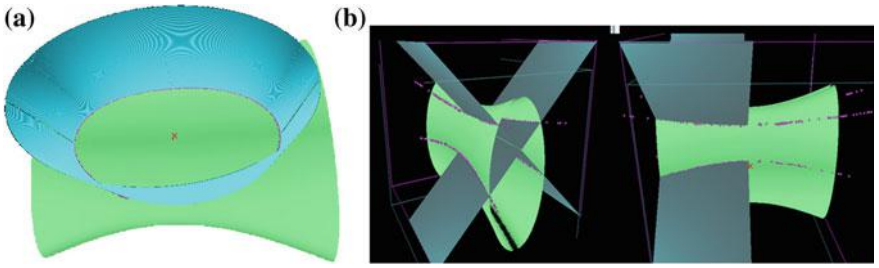


Fig. 7.2 **a** Intersection of two hyperboloids in $G_{6,3}$. **b** Intersection of two quadrics: two planes crossing the origin and a hyperboloid. Figures provided by Julio Zamora-Esquivel

In OPNS, curves lie in the intersection of the surfaces, e.g., the intersection of an ellipsoid and a plane is an ellipse. In general by intersecting two surfaces, one can compute very interesting contours which in turn can be used effectively for the applications as geometric constraints. The intersection or meet operation of two surfaces can be computed in OPNS via the wedge of the two surfaces

$$\mathcal{C}^* = S_1^* \wedge S_2^*. \tag{7.83}$$

Similarly as in Eq. (6.69), the ellipsoid $\boldsymbol{A}_r$ and the hyperplane $\boldsymbol{A}_{r+1}$ are related via the point $\boldsymbol{Q}$ (dual of the ellipsoid $\boldsymbol{Q}^*$); thus, we then rewrite Eq. (6.68) as follows:

$$\boldsymbol{Q} = \boldsymbol{A}_r \boldsymbol{A}_{r+1}^{-1}. \tag{7.84}$$

Using Eq. (7.84) and given the plane π (A_{r+1}) and the ellipse $\boldsymbol{q}$ ($\boldsymbol{A}_r$), we can compute the ellipsoid back

$$\boldsymbol{Q} = \boldsymbol{q}\pi^{-1}. \tag{7.85}$$

Figure 7.2 shows the intersection of two hyperboloids, and a quadric (two planes crossing the origin) and an hyperboloid. Now, using this result given the line $\boldsymbol{L}$ and the sphere $\boldsymbol{s}$ or an ellipsoid $\boldsymbol{Q}$, we can compute the IPNS form of the pair of points $\boldsymbol{P}\boldsymbol{P}$ lying on the sphere or on the ellipsoid

$$\boldsymbol{P}\boldsymbol{P} = \boldsymbol{s}\boldsymbol{L} = \boldsymbol{s} \wedge \boldsymbol{L}, \tag{7.86}$$

$$\boldsymbol{P}\boldsymbol{P} = \boldsymbol{Q}\boldsymbol{L} = \boldsymbol{Q} \wedge \boldsymbol{L}. \tag{7.87}$$

The OPNS of the pair the points is then given by

$$\boldsymbol{P}\boldsymbol{P}^* = \boldsymbol{x}_{w_1} \wedge \boldsymbol{x}_{w_2}. \tag{7.88}$$

7.3.4 Transformations of $G_{6,3}$

In Sect. 3.7.3, we study the projective groups of $G_{3,3}$ using versors; these transformations can be used for the projective transformation of a line $\boldsymbol{L} \in G_{3,3}$ or $\boldsymbol{L} = \boldsymbol{n} + e_\infty \boldsymbol{m}\, in G_{6,3}$.

Now, similar as in the case of conformal geometric algebra $G_{4,1}$, for the geometric objects in $G_{6,3}$, we can use the transversion $\boldsymbol{K_b}$ in terms of the origin $e_0 = e_{01} + e_{02} + e_{03}$, the translator $\boldsymbol{T}$ in terms of the point at infinity $e_\infty = \frac{1}{3}(e_{\infty 1} + e_{\infty 2} + e_{\infty 3})$, the rotor using a bivector $\boldsymbol{n}$, and the their combination for the motor $\boldsymbol{M}$ in terms of the screw axis $\boldsymbol{L}$, namely,

$$\begin{aligned}
\boldsymbol{K_b} &= 1 + \mathbf{b} e_o, \\
\boldsymbol{T} &= e^{-\frac{t}{2} e_\infty} = 1 + \frac{1}{2} \boldsymbol{t} e_\infty, \\
\boldsymbol{R} &= e^{-\frac{\theta}{2} \boldsymbol{n}} = \cos\left(\frac{\theta}{2}\right) - \sin\left(\frac{\theta}{2}\right) \boldsymbol{n}, \\
\boldsymbol{M} &= \boldsymbol{T}\boldsymbol{R} = e^{-\frac{\theta}{2} \boldsymbol{L}}.
\end{aligned} \tag{7.89}$$

The versor for perspectivity called $perspector$ can be computed either using the bivectors $e_i e_j$ like Eq. (3.168) or following the similar approach using the bivectors $e_{0i} \wedge e_{\infty j}$ like Eq. (3.169) to get

$$\begin{aligned}
\boldsymbol{P} &= e^{\boldsymbol{B}_f} = e^{\frac{f_3 e_4 e_8 - f_2 e_4 e_9 + f_1 e_5 e_9}{2}}, \\
&= e^{\frac{f_3 e_{01} \wedge e_{02} - f_2 e_{01} \wedge e_{03} + f_1 e_{02} \wedge e_{03}}{2}}, \\
&= 1 + \frac{1}{2}(f_3 e_4 e_8 - f_2 e_4 e_9 + f_1 e_5 e_9), \\
&= 1 + \frac{1}{2}(f_3 e_{01} \wedge e_{02} - f_2 e_{01} \wedge e_{03} + f_1 e_{02} \wedge e_{03}).
\end{aligned} \tag{7.90}$$

The versor for the Lorentz transformation called $Lorentor$ can be computed either using the bivectors $e_i \wedge e_j$ like Eq. (3.168) or following the similar approach using the bivectors $e_{0i} \wedge e_{\infty j}$ like Eq. (3.169) to get

$$\begin{aligned}
\boldsymbol{R}_L &= e^{\boldsymbol{B}_L}, \\
&= e^{\frac{(e_5 e_6 - e_8 e_9) + (e_4 e_6 - e_7 e_9) + (e_4 e_5 + e_7 e_8)}{2}}, \\
&= e^{\frac{(e_{02} \wedge e_{\infty 3} - e_{02} \wedge e_{\infty 3}) + (e_{\infty 1} \wedge e_{03} - e_{01} e_{\infty 3}) + (e_{\infty 1} \wedge e_{02} - e_{01} \wedge e_{\infty 1})}{2}}.
\end{aligned} \tag{7.91}$$

The versor for the shear transformation called $sheartor$ can be computed either using the bivectors $e_i e_j$ like Eq. (3.168) or following the similar approach using the bivectors $e_{0i} \wedge e_{\infty j}$ like Eq. (3.169) to get

$$\begin{aligned}\boldsymbol{S} &= e^{\boldsymbol{B}_S} \\ &= e^{\frac{a_{23}(e_5e_6-e_9e_8)+a_{13}(e_4e_6-e_9e_7)+a_{12}(e_4e_5-e_8e_7)}{2}} \\ &\qquad e^{-\frac{a_{32}(e_6e_5-e_8e_9)+a_{31}(e_6e_4-e_7e_9)+a_{12}(e_4e_5-e_8e_7)}{2}}, \\ &= e^{\frac{a_{23}(e_{\infty 2}\wedge e_{03}-e_{02}\wedge e_{\infty 3}+a_{13}(e_{\infty 1}\wedge e_{03}-e_{01}e_{\infty 3})+a_{12}(e_{\infty 1}\wedge e_{02}-e_{01}\wedge e_{\infty 2})}{2}} \\ &\qquad e^{\frac{a_{32}(e_{\infty 3}\wedge e_{02}-e_{03}\wedge e_{\infty 2})+a_{31}(e_{\infty 3}\wedge e_{01}-e_{03}e_{\infty 1})+a_{21}(e_{\infty 2}\wedge e_{01}-e_{02}\wedge e_{\infty 1}}{2}}.\end{aligned} \tag{7.92}$$

For example, we can apply an affine transformation to a cube C and stretch two faces to rhomboids and the other four faces to rectangles as follows:

$$\mathcal{R} = \boldsymbol{S}\mathcal{C}\tilde{\boldsymbol{S}} \tag{7.93}$$

By considering the fact that the point at infinity is a linear combination of three points at infinity in general position $e_\infty = \frac{1}{3}(e_{\infty 1} + e_{\infty 2} + e_{\infty 3})$, we can formulate the dilator D to scale or dilate a surface in the direction pointed by each of these points at infinity; thus, the versor for the dilation transformation $\boldsymbol{D}$ can be formulated as in $G_{3,3}$ either using the bivectors e_ie_j like Eq. (3.168) or following the similar approach using the bivectors $e_0i \wedge e_{\infty j}$ like Eq. (3.169) to get

$$\begin{aligned}\boldsymbol{D} &= e^{\boldsymbol{B}_D}, \\ &= e^{\frac{(1/0,\pm)e_7e_4+(0/1,\mp)e_8e_5+(0/1,\mp)e_9e_6+(-1/0,\pm)e_4e_7(0/-1,\mp)e_5e_8+(0/-1,\mp)e_6e_9}{2}}, \\ &= e^{\frac{(1/0,\pm)e_{\infty 1}e_{01}+(0/1,\mp)e_{\infty 2}e_{02}+(0/1,\mp)e_{infty3}e_{03}}{2}} \\ &\qquad e^{\frac{(-1/0,\pm)e_{01}\wedge e_{\infty 1}+(0/-1,\mp)e_{02}\wedge e_{\infty 2}+(0/-1,\mp)e_{03}\wedge e_{\infty 3}}{2}}.\end{aligned} \tag{7.94}$$

For example, to transform an sphere S into an ellipsoid, we apply the dilator as follows:

$$Q = \boldsymbol{D}S\tilde{\boldsymbol{D}}. \tag{7.95}$$

7.4 The Geometric Subalgebras $G^+_{9,3}$ and $G^+_{6,0,6}$

We are interested to extend the double motor $G^+_{6,0,2}$ to $G^+_{6,0,6}$ in order to augment the null vectors to six and hence to have the necessary bivectors to formulate the affine transformation of lines in terms of versors. This will be a suitable mathematical framework to handle the algebra of lines, screws, and co-screws utilizing versors for $SE(3)$ Lie Group and for the affine transformations of lines.

Let us consider the geometric subalgebra $G^+_{9,3}$ which is generated as follows:

$$G^+_{9,3} = \mathtt{gen}\{1, e_ie_j, e_ie, f_if_j, f_if, \ldots, I\}, \tag{7.96}$$

where the bivectors are computed using an orthogonal set of basis vectors e_i, $f_i = e_{i+3}$ for $i = 1, 2, 3$ and six null vectors $e, f, g, \bar{e}, \bar{f}, \bar{g}$. $G^+_{9,3}$ has 66 bivectors like $e_ie_j, e_ie, f_if_j, f_if, e\bar{e}, f\bar{f}, g\bar{g}, \ldots, e_8e_9$ and some pseudoscalars like $I_e = e_1e_2e_3e$, $I_f = f_1f_2f_3f$, $I_g = g_1g_2g_3g$, and $I = e_1e_2e_3ef_1f_2f_3fg_1g_2g_3g = I_eI_fI_g$. Note that one can transform a set of three bivectors e_ie_j, e_ie to its orthogonal set f_if_j, f_if using the pseudoscalar $I_{ef} = I_eI_f$ or e_ie_j, e_ie to its orthogonal set g_ig_j, g_ig using the pseudoscalar $I_{eg} = I_eI_g$ or f_if_j, f_if to its orthogonal set g_ig_j, g_ig using the pseudoscalar $I_{fg} = I_fI_g$, namely,

$$\begin{aligned} f_if_j &= I_{ef}e_ie_j, \quad f_if = I_{ef}e_ie, \\ g_ig_j &= I_{eg}e_ie_j, \quad g_ig = I_{eg}e_ie, \\ g_ig_j &= I_{fg}f_if_j, \quad g_ig = I_{fg}f_if, \end{aligned} \tag{7.97}$$

for $i, j = 1, 2, 3$. According to Eq. (7.52), the set of null basis $G^+_{9,3}$ are

$$\begin{aligned} e_{01} &= \frac{(e_7 - e_{10})}{2}, \quad e_{\infty 1} = e_7 + e_{10}, \\ e_{02} &= \frac{(e_8 - e_{11})}{2}, \quad e_{\infty 2} = e_8 + e_{11}, \\ e_{03} &= \frac{(e_9 - e_{12})}{2}, \quad e_{\infty 3} = e_9 + e_{12}, \end{aligned}$$

with the properties

$$e^2_{0i} = e^2_{\infty i} = 0, \quad e_{\infty i} \cdot e_{0i} = -1. \tag{7.98}$$

We will rename them as

$$\begin{aligned} \bar{e} &= e_{01}, \quad e = e_{\infty 1}, \\ \bar{f} &= e_{02}, \quad f = e_{\infty 2}, \\ \bar{g} &= e_{03}, \quad g = e_{\infty 3}. \end{aligned} \tag{7.99}$$

The goal to extend of $G^+_{6,0,2}$ to $G^+_{6,0,6}$ is to get the double motor algebra but equipped with six null vectors $\bar{e}, \bar{f}, \bar{g}, e, f, g$. Note that $\bar{e}, \bar{f}, \bar{g}$ represent the null vectors lying at the plane crossing the origin and e, f, g three null vectors called points at infinity lying at the plane at infinity.

Now, the representations of the screws $\boldsymbol{s}$, co-screws and momenta $\mathcal{P}$ are equal as in $G^+_{6,0,2}$. Since, in $G^+_{6,0,6}$, we have three null vectors lying on null cone e, f, g and other three $\bar{e}, \bar{f}, \bar{g}$ on the dual null cone, we can formulate in terms of bivectors (generated by these null vectors) the affine transformation of the inertia matrix $\mathtt{N}$ as it is explained next. Since $G^+_{6,0,2}$ is a subalgebra of $G^+_{9,3}$, we can identify $G^+_{6,0,6}$ as the subalgebra of $G^+_{9,3}$ as well, consisting of bivectors in terms of six vector bases $e_1, e_2, e_3, e_4 = f_1, e_5 = f_2, e_6 = f_3$ and the six null vectors named as $\bar{e}, \bar{f}, \bar{g}, e, f, g$. Thus, we can use these six null vectors of $G^+_{6,0,6}$ to represent the inertia matrix $\mathtt{N}$. According to Eq. (7.19), the inertia matrix of a rigid body is given by a 6×6 matrix

$$\mathtt{N} = \begin{bmatrix} \mathtt{I} & m[\mathtt{c}]_\times \\ m[\mathtt{c}]^{\mathtt{T}}_\times & m\mathtt{I}_3 \end{bmatrix}, \tag{7.100}$$

where $\mathtt{I}$ is the 3×3 inertia matrix and $\mathtt{I}_3$ is the identity matrix, m is the mass of the body, and $\mathtt{c}$ is the position vector of the center of mass expressed as $[\mathtt{c}]_\times$ an antisymmetric matrix to compute its cross product. Since $G_{6,0,2}$ has not enough null vectors as in $G_{3,3}$, $G^+_{9,3}$ or $G^+_{6,0,6}$, we can now express $\mathtt{N}$ in $G^+_{6,0,6}$ as an affine transformation using bivectors as it is shown in Sect. 3.5.

Note that Eq. (7.100) corresponds to the line transformation into the projective space $\mathbb{P}^5$. Similarly, as shown in Sect. 3.7.3 for Lie groups in $\mathbb{R}^{3,3}$, Eq. (7.100) can be represented using the tables of the inner product of the unit basis or the null basis

$\cdot$	e_1	e_2	e_3	e_7	e_8	e_9
e_1	1	0	0	0	0	0
e_2	0	1	0	0	0	0
e_3	0	0	1	0	0	0
e_7	0	0	0	-1	0	0
e_8	0	0	0	0	-1	0
e_9	0	0	0	0	0	-1

$\cdot$	$\bar{e}$	$\bar{\mathtt{f}}$	$\bar{g}$	e	f	g
$\bar{e}$	0	0	0	1	0	0
$\bar{\mathtt{f}}$	0	0	0	0	1	0
$\bar{g}$	0	0	0	0	0	1
e	1	0	0	0	0	0
f	0	1	0	0	0	0
g	0	0	1	0	0	0

.

Using these tables, we can multiply the entries of the inertia matrix with bivectors as follows:

$$\mathtt{T}_{\mathtt{e}_1,\mathtt{e}_2,\mathtt{e}_3,\mathtt{e}_7,\mathtt{e}_8,\mathtt{e}_9} = \begin{bmatrix} i_{11} & i_{12} & i_{13} & 0 & -mc_3 & mc_2 \\ i_{21} & i_{22} & i_{23} & mc_3 & 0 & -mc_1 \\ i_{31} & i_{32} & i_{33} & -mc_2 & mc_1 & 0 \\ 0 & mc_3 & -mc_2 & m & 0 & 0 \\ -mc_3 & 0 & mc_1 & 0 & m & 0 \\ mc_2 & -mc_1 & 0 & 0 & 0 & m \end{bmatrix} \tag{7.101}$$

$$\mathtt{T}_{\bar{\mathtt{e}},\bar{\mathtt{f}},\bar{\mathtt{g}},\mathtt{e},\mathtt{f},\mathtt{g}} = \begin{bmatrix} 0 & -mc_3 & mc_2 & m & 0 & 0 \\ mc_3 & 0 & -mc_1 & 0 & m & 0 \\ -mc_2 & mc_1 & 0 & 0 & 0 & m \\ i_{11} & i_{12} & i_{13} & 0 & mc_3 & -mc_2 \\ i_{21} & i_{22} & i_{23} & -mc_3 & 0 & mc_1 \\ i_{31} & i_{32} & i_{33} & mc_2 & -mc_1 & 0 \end{bmatrix}. \tag{7.102}$$

Inertia and m$\mathtt{I}_3$: Similar as in Sect. 3.7.3, we can compute the Lie operator for the inertia $\mathtt{I}$ and $m\mathtt{I}_3$

$$\mathcal{L}_{\mathtt{I},\mathtt{mI}_3} = \begin{bmatrix} & 0 & & \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \\ \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} & & 0 \end{bmatrix}. \tag{7.103}$$

The versor for inertia I and $\mathtt{mI}_3$ called *inertor* can be computed either using the bivectors $e_i \wedge e_j$ as Eq. (3.168) or following the similar approach using the bivectors combining e, f, g, $\bar{e}$, $\bar{f}$, $\bar{g}$ as Eq. (3.169) to get

$$\begin{aligned} \boldsymbol{I}_I &= e^{\boldsymbol{B}_I}, \\ &= e^{-\frac{i_{11}e\wedge\bar{e}+i_{12}e\wedge\bar{f}+i_{13}e\wedge\bar{g}+i_{21}f\wedge\bar{e}+i_{22}f\wedge\bar{f}+i_{23}f\wedge\bar{g}+i_{31}g\wedge\bar{e}+i_{32}g\wedge\bar{f}+i_{33}g\wedge\bar{g}+m\bar{e}\wedge e+m\bar{f}\wedge f+m\bar{g}\wedge g}{2}}. \end{aligned} \tag{7.104}$$

$\mathbf{m}[c]_x$ and $\mathbf{m}[c]_x^T$: Similar as in Sect. 3.7.3, we can computed the Lie operator simultaneously for both $\mathrm{m}[c]_x$ and $\mathrm{m}[c]_x^T$

$$\mathcal{L}_{m[c]_x, m[c]_x^T} = \begin{bmatrix} \begin{bmatrix} 0 & -1 & 1 \\ 1 & 0 & -1 \\ -1 & 1 & 0 \end{bmatrix} & & 0 \\ & 0 & \begin{bmatrix} 0 & 1 & -1 \\ -1 & 0 & 1 \\ 1 & -1 & 0 \end{bmatrix} \end{bmatrix}. \tag{7.105}$$

The versor for $m[c]_x$ and $\mathrm{m}[c]_x^T$ can be computed either using the bivectors $e_i \wedge e_j$ as Eq. (3.168) or following the similar approach using the bivectors combining e, f, g, $\bar{e}$, $\bar{f}$, $\bar{g}$ as Eq. (3.169) to get

$$\begin{aligned} \boldsymbol{I}_{mc} &= e^{\boldsymbol{B}_{mc}}, \\ &= e^{m\frac{c_3\bar{e}\wedge\bar{f}-c_2\bar{e}\wedge\bar{g}+c_3\bar{f}\wedge\bar{g}-c_1f\wedge g+c_2e\wedge g+c_3e\wedge f}{2}}. \end{aligned} \tag{7.106}$$

Thus, the resulting versor for the inertia matrix called *inertor* is computed in terms of the last versors as follows:

$$\boldsymbol{N} = \boldsymbol{I}_I \boldsymbol{I}_{mc} = e^{\boldsymbol{B}_I} e^{\boldsymbol{B}_{mc}} = e^{\boldsymbol{B}_I + \boldsymbol{B}_{mc}}. \tag{7.107}$$

Having formulated the inertia matrix as a versor, we can now write in $G^+_{6,0,6}$ or $G^+_{9,3}$ the dynamic Eq. (7.29) entirely using versors, screws, and co-screws. Recalling Eq. (7.28)

$$\{\boldsymbol{s}, \mathcal{P}\} = \left\{ \begin{pmatrix} \boldsymbol{w} \\ \boldsymbol{v} \end{pmatrix}, \begin{pmatrix} \boldsymbol{q} \\ \boldsymbol{p} \end{pmatrix} \right\} = \begin{pmatrix} \boldsymbol{w} \times \boldsymbol{q} + \boldsymbol{v} \times \boldsymbol{p} \\ \boldsymbol{w} \times \boldsymbol{p} \end{pmatrix} \tag{7.108}$$

and the dynamic Eq. (7.29) which was written as follows:

$$\mathtt{N}\frac{d}{dt}\boldsymbol{s} + \{\boldsymbol{s}, N\boldsymbol{s}\} = \mathtt{N}\frac{d}{dt}\boldsymbol{s} + \{\boldsymbol{s}, \mathcal{P}\} = \boldsymbol{\tau}. \tag{7.109}$$

With the versor formulation of the inertia matrix, an affine transformation in $\mathcal{P}^6$, we can rewrite Eq. (7.109) entirely using versors, screws, and co-screws, namely,

$$\boldsymbol{N}\dot{\boldsymbol{s}}\tilde{\boldsymbol{N}} + \{\boldsymbol{s}, \boldsymbol{N}\boldsymbol{s}\tilde{\boldsymbol{N}}\} = \boldsymbol{N}\dot{\boldsymbol{s}}\tilde{\boldsymbol{N}} + \{\boldsymbol{s}, \mathcal{P}\} = \boldsymbol{\tau}. \tag{7.110}$$

7.5 Exercises

7.1 Prove the invariance of the inertia matrix expressed in Eq. (7.22).

7.2 Show the relation between the shuffle product of Eq. (7.24) and the generalized inner product given by Eqs. (2.55) and (2.56). How we can use the generalized inner product to get same result as the shuffle product yields?

7.3 Prove in $G^+_{6,0,2}$, the invariance of the inertia matrix N expressed in Eq. (7.22), but now in terms of bivectors $\boldsymbol{N}$ Eq. (7.33) and the transformation T as motors $\boldsymbol{M}$.

7.4 Prove that we can infer that the inertia, the screw, and co-screw must transform according to

$$\begin{aligned}
\boldsymbol{N} &\to (\boldsymbol{M}\boldsymbol{M}_f)\boldsymbol{N}(\widetilde{\boldsymbol{M}}_f\widetilde{\boldsymbol{M}}),\\
\boldsymbol{s} &\to (\boldsymbol{M}\boldsymbol{M}_f)\boldsymbol{s}(\widetilde{\boldsymbol{M}}_f\widetilde{\boldsymbol{M}}),\\
\mathcal{P} &\to (\boldsymbol{M}\boldsymbol{M}_f)\mathcal{P}(\widetilde{\boldsymbol{M}}_f\widetilde{\boldsymbol{M}}).
\end{aligned}$$

7.5 Prove that the equation in terms of quatrivectors

$$\begin{aligned}
(\boldsymbol{M}\boldsymbol{M}_f)\boldsymbol{N}(\widetilde{\boldsymbol{M}}_f\widetilde{\boldsymbol{M}}) = {} & (d_x + mt_y^2) f_1 f e_1 e + d_y f_2 f e_2 e + (d_z + mt_y^2) f_3 f e_3 e\\
& - mt_y f_3 f e_2 e_3 + mt_y f_1 f e_1 e_2 + mt_y f_1 f_2 e_1 e\\
& - mt_y f_2 f_3 e_3 e + m f_2 f_3 e_2 e_3 + m f_3 f_1 e_3 e_1 + m f_1 f_2 e_1 e_2
\end{aligned}$$

corresponds to the 6×6 inertia matrix

$$\begin{bmatrix}
d_x + mt_y^2 & 0 & 0 & 0 & 0 & mt_y\\
0 & d_y & 0 & 0 & 0 & 0\\
0 & 0 & d_z + mt_y^2 & -mt_y & 0 & 0\\
0 & 0 & -mt_y & m & 0 & 0\\
0 & 0 & 0 & 0 & m & 0\\
mt_y & 0 & 0 & 0 & 0 & m
\end{bmatrix}.$$

7.6 One extends $G_{4,1}$ to $G_{6,3}$ to get three null vectors as origins and three null vectors as three points at infinity. Show with equations, why these null vectors can transform spheres to ellipsoids.

7.7 Using Eq. 7.69 of the ellipsoid, one can derive the equation of the cylinder, namely, setting the ratio of axis z to infinity, we get

$$C = \frac{p}{a^2}e_1 + \frac{q}{b^2}e_2 + \frac{1}{2}\left(\frac{p^2}{a^2} + \frac{q^2}{b^2} - 1\right)e_\infty + \left(\frac{1}{a^2}e_{01} + \frac{1}{b^2}e_{02}\right).$$

Derive the equation of an hyperboloid depicted in Fig. 7.1b.

7.8 In $G_{6,3}$, compute with eClifford and Maple [8] the intersection of a cylinder with a hyperboloid. Plot with Maple your results.

7.9 In $G_{6,3}$, compute with eClifford and Maple [8] the intersection of a sphere with a hyperboloid. Plot with Maple your results.

7.10 In $G_{6,3}$, compute with eClifford and Maple [8] the intersection of a ellipsoid with a plane. Plot with Maple your results.

7.11 Compute with eClifford and Maple [8] the transformation of an ellipsoid using the following Versors:

$$\begin{aligned}
\boldsymbol{K}_{\mathbf{b}} &= 1 + \mathbf{b}e_o, \\
\boldsymbol{T} &= e^{-\frac{t}{2}e_\infty} = 1 + \frac{1}{2}\boldsymbol{t}e_\infty, \\
\boldsymbol{R} &= e^{-\frac{\theta}{2}\boldsymbol{n}} = \cos\left(\frac{\theta}{2}\right) - \sin\left(\frac{\theta}{2}\right)\boldsymbol{n}, \\
\boldsymbol{M} &= \boldsymbol{T}\boldsymbol{R} = e^{-\frac{\theta}{2}\boldsymbol{L}}.
\end{aligned} \tag{7.111}$$

Plot with Maple your results.

7.12 In $G_{6,3}$, compute with eClifford and Maple [8] the rigid motion of an hyperboloid and the transformation of a sphere to an ellipsoid. Plot with Maple your results.

7.13 In $G_{6,3}$, using eClifford and Maple [8] apply the Perspector to cube and a Lorentor to a cube. Plot with Maple and interpret your results the effect of the perspectivity and the transformation in space and time by the Lorentz transformation.

7.14 In $G^+_{9,3}$, using eClifford and Maple [8] apply the sheartor to a cube and a dilator to a cube. Plot with Maple and interpret your results.

7.15 In $G^+_{9,3}$, using eClifford and Maple [8] apply the dilator to transform a sphere into an ellipsoid. Plot with Maple and interpret your results.

7.16 In $G^+_{9,3}$, write the equations of a screw line $\boldsymbol{L}$, a twist $\boldsymbol{s}$, and using the inertor a momentum $\boldsymbol{p}$, then a wrench $\boldsymbol{w}$. Explain your self that these geometric entities are needed to write the Newton–Euler dynamic equation of serial manipulators.

7.17 In $G^+_{9,3}$, compute the Lie bracket $[\boldsymbol{s}_i, \boldsymbol{s}_j]$ of twists $\boldsymbol{s}_i$ and $\boldsymbol{s}_j$.

7.18 In $G^+_{9,3}$, compute the Lie co-bracket $\{\boldsymbol{s}_i, \boldsymbol{w}_j\}$ of twist $\boldsymbol{s}_i$ and wrench $\boldsymbol{w}_j$.

7.19 In $G^+_{9,3}$, write the direct kinematics of a n-DOF serial manipulator.

7.20 In $G^+_{9,3}$, write the direct kinematics of a four-leg Steward platform.

7.21 In $G^+_{9,3}$, using Eq. (7.110), write of a Newton–Euler procedure to compute the dynamics of a double spherical pendulum.

7.22 In $G^+_{9,3}$, using Eq. (7.110), write of a Newton–Euler procedure to compute the dynamics of a robot manipulator with three rotational joints.

References

1. Bayro-Corrochano, E., Daniilidis, K., & Sommer, G. (2000). Motor algebra for 3D kinematics. The case of the hand–eye calibration. *International Journal of Mathematical Imaging and Vision, 13*(2), 79–99.
2. Bayro-Corrochano, E. (2001). *Geometric computing for perception action systems*. New York: Springer Verlag.
3. Selig, J., & Bayro-Corrochano, E. (2010). Rigid body dynamics using Clifford algebra. *Advances in Applications of Clifford algebras*, *20*, 141–154.
4. Eaester, R. B., & Hitzer, E. (2017). Double conformal geometric algebra. *Advances in Applied Clifford Algebras*, *27*, 2175–2199.
5. White, N. (1994). Grassmann-Cayley algebras in robotics. *Journal of Intelligent Robot Systems*, *11*, 97–187.
6. Zamora-Esquivel, J. (2014). $G_{6,3}$ Geometric algebra: Decription and implementation. *Advances in Applied Clifford Algebras*, *24*, 493–514.
7. Li, H., Hestenes, D., & Rockwood, A. (2001). Generalized homogeneous coordinates for computational geometry. In G. Sommer (Ed.), *Geometric computing with Clifford algebra* (pp. 27–59). New York: Springer-Verlag.
8. Ablamowicks, R. eCLIFFORD Software packet using Maple for Clifford algebra. Computations. http://math.tntech.edu/rafal.

Chapter 8
Programming Issues

> "A computer would deserve to be called intelligent if it could deceive a human into believing that it was human" (Turing Test) in Computing Machinery and Intelligence (1950).
> Alan Mathison Turing (6.23.1912–6.7.1954)

In this chapter, we will discuss the programming issues to compute in the geometric algebra framework. We will explain the technicalities for the programming which you have to take into account to generate a sound source code. At the end, we will discuss the use of specialized hardware as FPGA and Nvidia CUDA to improve the efficiency of the code processing for applications in real time.

8.1 Main Issues for an Efficient Implementation

In the last decade, there were many attempts to develop software for Clifford algebra and geometric algebra computing some for computer science and numerical computing [1–3] and others for symbolic computing [4–6]. The efficient implementation is still an issue which depends greatly on the algorithmic complexity of the required algorithms and also the kind of accelerating cards being used. Fast hardware for running geometric algebra algorithms were recently proposed by [7–10]. Nevertheless, it is still a long way to go in order to have an efficient and fast software to run geometric algorithms for theorem proving and for real-time applications. We have to admit, however, that the current developments are good contributions to achieve this main goal.

According to the majority of the Clifford or geometric algebra software developers, it seems that there are already three well-established issues for an efficient implementation: the basic computing entities or multivectors, the role of metric, and the computational burden caused by the number of involved basic operations. We will next discuss these issues from our point of view and including useful suggestions of other authors [11, 12].

© Springer Nature Switzerland AG 2020

E. Bayro-Corrochano, *Geometric Algebra Applications Vol. II*,
https://doi.org/10.1007/978-3-030-34978-3_8

1. Multivectors: The geometric entities of a geometric algebra G_n are represented as multivectors which are expanded in a multivector basis of length 2^n. It is not prudent to represent all multivectors in a frame including many zeros, rather specialize the representation fixing the length to accomplish the needed representation and geometric products ignoring all the unnecessary multivector components. This implies a more limited use of grades or basis blades which consequently reduces the storage and processing time.
2. When we formulate representations and algorithms for a certain problem, we need to identify exactly the appropriate geometric algebra with a suitable metric. This is not necessarily a trivial task; to answer the question in which metric space $\mathbb{R}^{m,n}$ we should work, it will depend how well we are acquainted with the problem in question and the manner how we should tackle the problem. For instance, when we work in classification using neural computing, it is more advantageous to treat the feature space in higher dimensions in order to linearize the classification problem. Thus, we should not develop a program for a general geometric algebra and apply it for a particular problem. All indicates that we should have a specialized implementation for every metrically different geometric algebra.
3. The number of basic operations on multivectors compared with those in vector calculus (linear transformations, inner or cross products, norms, etc.) is quite large. Usual multivector expressions demand of multiple products and operations which can be reduced in number and complexity by folding them into one calculation instead to execute them one by one by calling a series of generic functions. This tells us that we should rather program things out explicitly for each operation restricted to each multivector type. It is appropriate to avoid a combinatorial explosion due to the number of multivector blade components and variables.

The analysis of these three issues suggests us not to implement a program for a general geometric algebra rather in a metric-defined geometric algebra; we have to break it up and specialize both the representation and the operation computations. Bear in mind that multivectors are unfortunately too big, the role of metrics constraints indeed the world representation but not necessarily enough for an efficient computation.

The operations on multivectors are, in general, simple and universal but unfortunately too slow if they are not specialized, optimized, and formulated in such a way that one can also take advantage of cost-effective hardware to speed up the computations.

8.1.1 Specific Aspects for the Implementation

The specific considerations for an efficient implementation can be summarized as follows:

- Do not waste computer memory. Try to store each variable as compactly as possible. Keep circumstantial results of functions in between which can reused later on.

- Implement the generic functions over the algebra in a efficient manner, such as factorization, meet, etc. For that, one should consider various practical details:

(i) Reduce unpredictable memory access.
(ii) Similar with the FFT, find zero coordinates and avoid to process them.
(iii) Optimize nontrivial geometric algebra expressions, particularly being careful not to implement them as a series of function calls.
(iv) Conditional branches are prohibitive in order to avoid the loss of processor cycles due to an incorrectly predicted conditional branch.
(v) Whenever possible, unroll loops, that is, let them run only over the dimension of the algebra or the subspaces determined by grade of the blade of the k-vectors.

8.2 Implementation Practicalities

As you may have noticed, we have not mentioned a particular programming language. You should consider as potential programming languages C++, JAVA, Python, MATLAB, and for symbolic calculations Mathematica and Maple with eClifford [4]. In the last section, we give some comments of some existing multivector software packets.

In general, you should consider the specification of the algebra to work in, the implementation of the general multivector class, and the generation of optimized functions. Next, we discuss these aspects in some detail.

8.2.1 Specification of the Geometric Algebra $G_{p,q}$

The program requires first to know

- the dimension n of the algebra G_n;
- the metric of the algebra $G_{p,q}$, type of quadratic form expressing the space signature;
- The definition of the specialized types, and list of basis blades and versors;
- The definition of the constants. This is referred to as constant multivector values, which should be considered constant, for example, basis vectors, e_0 and the pseudoscalar I_n. By including the constants in the algebra specifications, they can be easily recognized and optimized away by the code generator or compiler.

8.2.2 The General Multivector Class

For the program, the implementation of a general multivector is more appropriate. This is because many equations of geometric algebra are computed regardless of the multivector type of the input, and expansions and contractions vary the output multivector length as well. On the other hand, due to noisy entries, the multivector type of variables may change unexpectedly at run time.

We should care for the administration of the coordinates of the general multivector, because during the computation we can naively try to store all of the 2^n multivector components, some of which could in fact be zero or perhaps not all are needed to accomplish certain computational goal. Therefore, in order to increase the computational performance, we should avoid to store coordinates which are zero. Paying attention to the grade of blades, we can select only the k-blades of a multivector which necessarily are involved in a certain computation, rejecting the irrelevant ones. Since versors are even order odd, we can at least leave the half of their grade counterparts. Note that for low-dimensional spaces, selecting the involved k-vectors reduces the dependency of a slow conditional branching by looking at the coordinate level. However, reduction by grade still misses some good opportunities at higher level of complexity due to the inherent characteristics of the computation. This applies to entities which have common characteristics for intervening in the computation and depend heavily on the specific use of the algebra, for example, select only flat points in conformal geometric algebra. In general, alerting the processor with some proper constraints of the problem in question could help to group zero coordinates of blades of the same grade, and therefore alleviate even further the problem caused by unnecessary computations. Symbolic simplification and factorization of multivector functions will help to reduce their processing time, particularly if they are called in loops.

For each multivector type used in the program, it is desirable to generate specialized multivector classes, bearing in mind that the class provides storage for the non-constant coordinates and some functionality to convert back and forth between the general- and the specialized multivector classes. The rest of the functionality will be provided by the functions over the entire algebra.

8.2.3 Optimization of Multivector Functions

The optimization of multivector functions over elements of the algebra is extremely important to speed up the processing and to guarantee high performance of the computer program. To start, the functions are formulated based on their high-level definitions. However, the precise syntax of the definition depends greatly on the generative programming method used. Using programming language C++ a kind of *meta-programming approach* the functions are formulated as a C++ template functions. These functions are instantiated with specific type of multivectors. A reasonable procedure for generating code and possible optimized from a high-level definition should take into consideration the following common-sense advices:

- The types of the function arguments are substituted with efficient specializations.
- The expressions in the function are formulated explicitly on a basis.
- Simplify the expressions symbolically, so that most of the operations and products are executed at the basis level; unnecessary computations are removed; identical terms are fused; reduce length looking at coordinate level, grade of k-vectors, and at group level of complex geometric entities, etc.

– Avoid unnecessary branching, conditionals, and loops. For multivector classification and factorization, it is advisable to unroll loops in order to ensure that each variable has a fixed multivector type. Here, via blade factorization after loops are unrolled, all variables in the algorithm get a fixed type; consequently, most conditional branches can be removed.
– The algorithms for meet and join of blades cannot be optimized by using specialized multivectors. This is because in the algorithm the multivector type of several variables is normally not fixed. Also, there are several other unavoidable conditional branches.
– The return type of the function is specified.

8.2.4 Factorization

Many researchers have explored ways to factorize representations [2, 11, 13]. Common multiplicative representations in geometric algebra include blades as wedge products of vectors and versors as geometric product of vectors (reflections for rotations, reversions, translations, etc). A k-blade corresponds to the wedge product of k-vectors, and similarly a versor is the result of the wedge of certain number n of vectors. After factorization of a k-blade or a versor, for both, the resulting amount of vectors can be stored as a list of vectors. Note that the storage requirements of blades and versors become $O(n^2)$ instead of $O(2^n)$, when we factorize them using the basis-of-blades standard method. Recall that multivectors are often not homogeneous, i.e., k-blades or versors are expanded as a sum of multiple blades or versors of different grades; thus, this requires more storage.

As an illustration, we include the following procedure for factorizing a blade which was proposed by [2, 11, 13]:

- Input: A nonzero-blade $\boldsymbol{X}$ of grade r.
- Compute its norm $\alpha = |\boldsymbol{X}|$.
- For $\boldsymbol{X}$, find the basis blade $\check{\boldsymbol{X}}$ with the largest coordinate and then determine the r basis vectors e_i that span $\check{\boldsymbol{X}}$.
- Normalize the input blade $\boldsymbol{U} = \boldsymbol{X}/\alpha$.
- For all the r basis vectors $e_i \in$ of $\check{\boldsymbol{X}}$ compute:
  - Project e_i onto $\boldsymbol{U}$, i.e., $\boldsymbol{g}_i = (e_i \cdot \boldsymbol{U})\boldsymbol{U}^{-1}$.
  - Normalize $\boldsymbol{g}_i$ and store it into the list of factors.
  - Update $\boldsymbol{U} \leftarrow \boldsymbol{g}_i^{-1} \cdot \boldsymbol{U}$.
- Obtain the last factor $\boldsymbol{g}_r = \boldsymbol{U}$ and normalize it.
- Output: The list of the factors $\boldsymbol{g}_i$ and the scale α.

Further research work has to be done for more efficient factorization methods specially for high-dimensional ($n < 10$) algebras. New insights in this regard can be found in Fontjine's Ph.D. thesis [13].

8.2.5 Speeding up Geometric Algebra Expressions

To reduce the complexity of multivector expressions, we can resort to a gradual symbolic simplification which can be then let run using cost-effective hardware, see Fig. 8.1. The main idea here is that after we formulated an algorithm in geometric algebra terms for a certain task, we reduce its complexity using symbolic simplification Maple with eClifford [4]. This produces a generic Intermediate Representation (IR) that can be used for the generation of different output formats such as C-code, FPGA descriptions (Verilog language), or CLUCalc code in order to visualize the results. For this purpose, an architecture called *Gaalop* was developed by Hildebrand et al. [14]. As an example, let us consider a problem of inverse kinematic. Figure 8.2 shows the rotation planes of robot arm elbow. We consider a circle which is the intersection of two reference spheres $z = s_1 \wedge s_2$. We compute then the pair of points (PP) as the meet of the swivel plane and the circle: $PP = z \wedge \pi_{swivel}$, so that we can choose a realistic position of the elbow. This is a point which lies on the meet of two spheres. This position is given by the 3-D coordinates p_{ex}, p_{ey} and p_{ez}. After the optimization of the equation of the inverse kinematics, we get an efficient representation for the computing of this elbow position point. You can see for p_{ex} in Fig. 8.3 its data flow and its pipeline structure. Using this implementation, the authors reported for the whole inverse kinematics algorithm a remarkable acceleration of more than 130 times faster than the speed of an unoptimized algorithm [15].

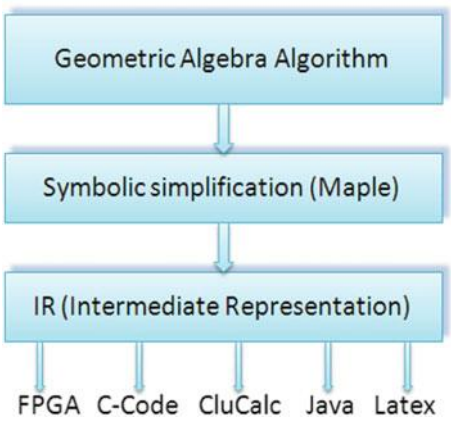


Fig. 8.1 Computing the pair of points as the meet of the swivel plane and the circle

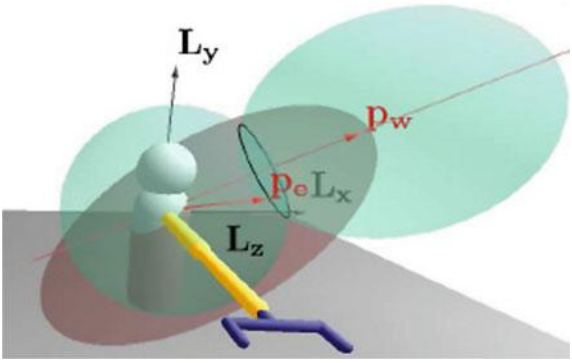


Fig. 8.2 Computing the pair of points as the meet of the swivel plane and the circle

8.2.6 Multivector Software Packets

The existing software packets for multivector programming are as follows:
Fortran-based CLICAL developed by the group of Pertti Lounesto. It is very useful for fast computation and theorem proving. It is available at
http://users.tkk.fi/ppuska/mirror/Lounesto/CLICAL.htm.

MATLAB-based geometric algebra tutorial called GABLE supports $N \leq 3$, and it is available at
http://staff.science.uva.nl/~leo/GABLE/index.html.

Maple-based eClifford for computing in any dimension, because the computing of the Clifford product is carried out using Welsh functions [1]. It is very useful for symbolic programming and theorem proving, and it is available at
http://math.tntech.edu/rafal.

C++-based CLUCAL. It is very useful for practicing and learning geometric algebra computations in 2D and 3D particularly for visualization, computer vision, and crystallography. It is available at
http://www.perwass.de/cbup/clu.html.

GAIGEN2 generates fast C++ or JAVA sources for low-dimensional geometric algebra. It is very useful for practicing and learning geometric algebra computing, and to try a variety of problems of computer science and graphics. It is available at
http://www.science.uva.nl/ga/gaigen/.

C++ MV 1.3.0 sources supporting $N \leq 63$. The author Ian Bell has developed up to 1.6 with significant functionality extensions and bug fixes. It is very powerful multivector software for applications in computer science and physics. It is available at
http://www.iancgbell.clara.net/maths/index.htm.

The C++ GEOMA v1.2 developed by Patrick Stein contains C++ libraries for Clifford algebra with orthonormal basis. It is available at

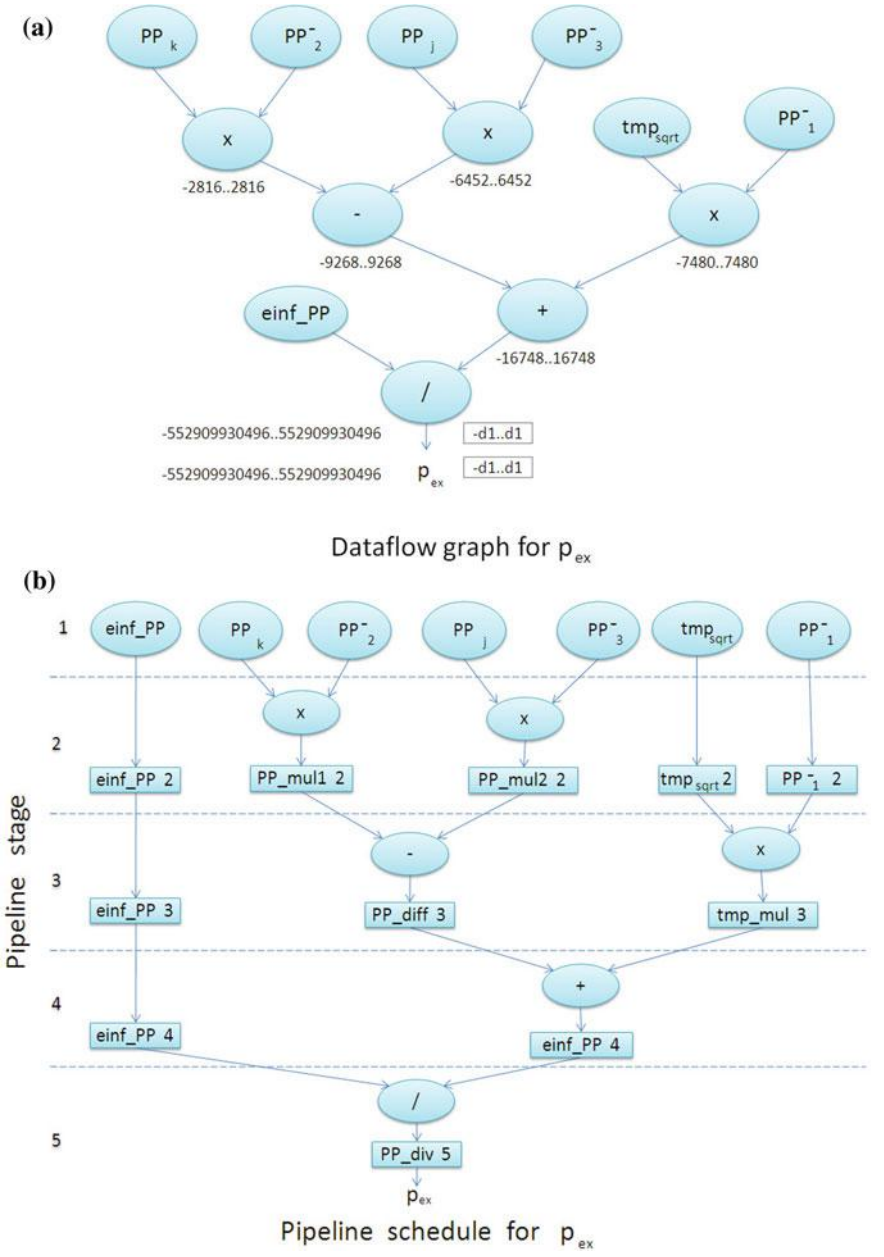


Fig. 8.3 Data flow and pipeline structure for a fast computing of the elbow position point

http://nklein.com/software/geoma.

The reader can also download our C++ programs which are being routinely used for applications in robotics, image processing, wavelets transforms, computer vision, neural computing, and medical robotics. These are available at
http://www.gdl.cinvestav.mx/edb/GAprogramming.

Readers, who want to develop their own programs for Clifford or geometric algebra applications, should consider the advice given in this section, learn from the above-cited multivector software packets, and adopt and integrate their new developments. To learn geometric algebra CLUCAL, GAIGEN2 and eClifford [4] are highly recommended. To write a C++ geometric algebra program, one should start to look the GEOMA, the MV 1.3.0, and the code generator of GAIGEN or download the programs of our homepage. For symbolical computing and proof of theorems, CLICAL [1] and eClifford [4] are extremely useful.

8.2.7 Specialized Hardware to Speed up Geometric Algebra Algorithms

There are many works using dedicated hardware or FPGA to accelerate geometric algebra algorithms:

- Dedicated hardware [16, 17].
- The use of FPGA for geometric algebra computing [10, 18].
- The use of Nvidia CUDA for geometric algebra computing.

In [19, Sect. 16.5], we illustrate the implementation of computer vision algorithms using conformal geometric algebra and speed up using FPGA. Currently, we are accelerating interpolation in conformal geometric algebra using FPGA and Nvidia CUDA.

References

1. Lounesto, P. (1997). *Clifford algebras and spinors*. Cambridge: Cambridge University Press.
2. Dorst, L., Fontjine, D., & Mann, T. GAIGEN 2: Generates fast C++ or JAVA sources for low dimensional geometric algebra. http://www.science.uva.nl/ga/gaigen/.
3. Perwass, C. B. U. (2006). CLUCal. http://www.clucal.info/.
4. Ablamowicks, R. eCLIFFORD Software packet using Maple for Clifford algebra. Computations. http://math.tntech.edu/rafal.
5. Ashdown, M. A. J. (1998). Maple code for geometric algebra. http://www.mrao.cam.ac.uk/~maja.
6. Dorst, L., Mann, S., & Bouma, T. (1999). GABLE: A Matlab tutorial for geometric algebra. http://www.carol.wins.uva.nl/~gable.

7. Perwass, C., Gebken, C., & Sommer, G. (2003). Implementation of a Clifford algebra co-processor design on a field programmable gate array. In R. Ablamowicz, (Ed.), *Clifford algebras: Application to mathematics, physics, and engineering*. *6th International Conference on Clifford Algebras and Applications*, Cookeville, TN (pp. 561–575). Progress in Mathematical Physics. Boston: Birkhäuser.
8. Mishra, B., & Wilson, P. (2005). Hardware implementation of a geometric algebra processor core. In *Proceedings of IMACS International Conference on Applications of Computer Algebra*, Nara, Japan. http://eprints.ecs.soton.ac.uk/10957/.
9. Gentile, A., Segreto, S., Sorbello, F., Vassallo, G., Vitabile, S., & Vullo, V. (2005). CliffoSor, an innovative FPGA-based architecture for geometric algebra. In *Proceedings of 45th Congress of the European Regional Science Association (ERSA)* (pp. 211–217), Vrije, Amsterdam, August 23–27.
10. Soria-García, G., Altamirano-Gómez, G., Ortega-Cisneros, S., & Bayro-Corrochano, E. (2017). FPGA implementation of a geometric voting scheme for the extraction of geometric entities from images. *Advances in Applications of Clifford Algebras*, *27*, 685–705.
11. Bell, I. C++ MV 1.3.0 to 1.6 sources supporting $N \leq 63$. http://www.iancgbell.clara.net/maths/index.htm.
12. Dorst, L., Fontjine, D., & Mann, S. (2007). *Geometric algebra for computer science. An object-oriented approach to geometry*. Morgan Kaufmann Series in Computer Science, Cambridge, MA.
13. Fontijne, D. (2007). *Efficient implementation of geometric algebra*. Ph.D. Thesis, University of Amsterdam. http://www.science.uva.nl/~fontjine/phd.html.
14. Hildenbrand, D., Pitt, J., & Koch, A. (2009) High-performance geometric algebra computing using Gaalop. In E. Bayro-Corrochano, G. Sheuermann, (Eds.), *Geometric algebra computing for engineering and computer science* (pp. 477–494). London: Springer.
15. Hildenbrand, D., Pitt, J., & Koch, A. (2010). Gaalop high performance parallel computing based on conformal geometric algebra. In E. Bayro-Corrochano, G. Sheuermann, (Eds.), *Geometric algebra computing for engineering and computer science* (Chap. 22, pp. 477–494). Springer-Verlag.
16. Sommer, G., & Perwass, C. (2004). Implementation of a Clifford algebra co-processor design on a field-programmable gate array. In *Clifford algebras: Applications to mathematics, physics, and engineering*. Progress in Mathematical Physics.
17. Franchini, S., Gentile, A., Grimaudo, M., Hung, C., Impastato, S., Sorbello, F., Vassallo, G., & Vitabile, S. (2007). A sliced coprocessor for native Clifford algebra operations. In *Proceedings of the 10th IEEE Euromicro Conference on Digital System Design–Architectures, Methods and Tools, DSD'07*, Lübeck (pp. 436–439).
18. Soria-García, G., Altamirano-Gómez, A., Ortega-Cisneros, S., & Bayro-Corrochano, E. (2017). Conformal geometric algebra voting scheme implemented in reconfigurable devices for geometric entities extraction. *IEEE Transactions on Industrial Electronics*.
19. Bayro-Corrochano, E. (2018). *Geometric algebra applications vol. I computer vision, graphics and neurocomputing*. Springer-Verlag.

Part II
Interpolation, Kinematics, and Dynamics

Chapter 9
Rigid Motion Interpolation

This chapter presents the motor interpolation and it is based on our previous works [1, 2]. We will use this technique when we interpolate geometric objects like points, lines, planes, circles, and spheres. Their equations formulated in conformal geometric algebra in Chap. 6 and summarized in Table 6.1 are included here again

$$x_c = x_e + \frac{1}{2}x_e^2 e_\infty + e_0, \tag{9.1}$$

$$\boldsymbol{L} = nI_E - e_\infty m = (\boldsymbol{n} + e_\infty m), \tag{9.2}$$

$$\pi = nI_E - e_\infty d = \boldsymbol{n} - e_\infty d, \tag{9.3}$$

$$s = p_c - \frac{1}{2}\rho^2 e_\infty = p_e + \left(\frac{p_e^2 - \rho^2}{2}\right) e_\infty + e_0. \tag{9.4}$$

9.1 The Motor Spherical Linear Interpolation Function

This interpolation method is a generalization of the well-known Spherical Linear Interpolation (SLERP) method [3]. Given $\boldsymbol{M}_i$ and $\boldsymbol{M}_f$, two motors representing the initial and final poses of a rigid body, respectively, the Motor Spherical Linear Interpolation function (MSLERP) is defined as follows:

$$MSLERP(t; \boldsymbol{M}_i, \boldsymbol{M}_f) = \boldsymbol{M}_i(\boldsymbol{M}_i^{-1}\boldsymbol{M}_f)^t, \tag{9.5}$$

with $t \in [0, 1]$. The underlying kinematic concept of the screw motion was explained in previous sections. A kinematic explanation of this interpolation method was given by [3, 4]. Since $(\boldsymbol{M}_i^{-1}\boldsymbol{M}_f)$ represents the finite screw motion between the initial and

© Springer Nature Switzerland AG 2020

E. Bayro-Corrochano, *Geometric Algebra Applications Vol. II*,
https://doi.org/10.1007/978-3-030-34978-3_9

final pose of the rigid body, specifically the product

$$(\boldsymbol{M}_i^{-1}\boldsymbol{M}_f)^t = \cos\left(t\left(\frac{\theta}{2}+e_\infty\frac{\boldsymbol{t}}{2}\right)\right)+\boldsymbol{L}\sin\left(t\left(\frac{\theta}{2}+e_\infty\frac{\boldsymbol{t}}{2}\right)\right),$$
$$= \cos\left(t\left(\frac{\hat{\theta}}{2}\right)\right)+\boldsymbol{L}\sin\left(t\left(\frac{\hat{\theta}}{2}\right)\right), \tag{9.6}$$

defines a screw motion of a dual angle $t\hat{\theta}$ along the screw axis $\boldsymbol{L}=\boldsymbol{n}+e_\infty\boldsymbol{m}$. In this equation, θ and $\boldsymbol{t}$ stand for the rotation angle and the translation bivector, respectively.

The Motor Linear Blending interpolation method (MLB) is defined as follows:

$$MLB(t;\boldsymbol{M}_i,\boldsymbol{M}_f)=\frac{(1-t)\boldsymbol{M}_i+t\boldsymbol{M}_f}{||(1-t)\boldsymbol{M}_i+t\boldsymbol{M}_f||}. \tag{9.7}$$

This procedure can be extended to the interpolation involving several poses:

$$MLB(t;\boldsymbol{M}_1,\boldsymbol{M}2)=\frac{w_1(t)\boldsymbol{M}_1+\cdots+w_n(t)\boldsymbol{M}_n}{||w_1(t)\boldsymbol{M}_1+\cdots+w_n(t)\boldsymbol{M}_n||}. \tag{9.8}$$

These weights are assumed to be convex, namely, $w_i\geq 0$ and $\sum_{i=1}^n w_i=1$. One can choose weights that are made coincident with the coefficients of the classical Lagrange's interpolating polynomials

$$w_j(t)=\prod_{k=1,k\neq j}^{n}\frac{t-t_k}{t_j-t_k}. \tag{9.9}$$

There are alternatives for choosing the t value [5]. A simple idea is to select t that coincides with the Euclidean distance between the coordinates of the origins of the moving Cartesian coordinate frame in the different poses.

If in the MLB algorithm the lengthy normalization is avoided, then the resulting motor does not necessarily belong to the hypersphere. Kavan et al. [4] proposed a refinement of the MLB, the Dual Quaternion Iterative Blending DIB algorithm; in this work, we reformulate the DIB using motors and it be called Motor Iterative Blending (MIB). Note, Algorithm 1 uses the resulting bivector $\boldsymbol{B}$ (se(3) Lie algebra) to compute

Algorithm 1 MIB: Motor Iterative Blending

procedure (MIB) $(\boldsymbol{M}_1,\ldots,\boldsymbol{M}_n,t)$
$\boldsymbol{M}_B = MLB(\boldsymbol{M}_1,\ldots,\boldsymbol{M}_n,t)$
`repeat`
$\boldsymbol{B}=\sum_{i-1}^{n} w_i\ log(\boldsymbol{M}_B^{-1}\boldsymbol{M}_i)$
$\boldsymbol{M}_B=\boldsymbol{M}_B e^{\boldsymbol{B}}$
`until` $||\boldsymbol{B}||\leq h$
`end procedure`

the motor (SE(3) Lie group). To interpolate geometric objects $\boldsymbol{O}$ in the conformal geometric algebra framework, represented in IPNS, we use Eqs. (9.1)–(9.4) of points, lines, planes, and Spheres, respectively, and apply the blending interpolated motor $\boldsymbol{M}_B$ and for spheres a Dilator $\boldsymbol{D} = e^{\ln \rho}$ as well,

$$x_c^t = \boldsymbol{M}_B^t x_c^{t-1} \tilde{\boldsymbol{M}}_B^t, \tag{9.10}$$

$$L^t = \boldsymbol{M}_B^t L^{t-1} \tilde{\boldsymbol{M}}_B^t, \tag{9.11}$$

$$\pi^t = \boldsymbol{M}_B^t \pi^{t-1} \tilde{\boldsymbol{M}}_B^t, \tag{9.12}$$

$$s^t = \boldsymbol{D}^t \boldsymbol{M}_B^t s^{t-1} \tilde{\boldsymbol{M}}_B^t \boldsymbol{D}^{-1^t}. \tag{9.13}$$

9.2 Study Quadric Interpolation Algorithm

Gfrerrer presents in [6] an interpolation algorithm based on the Study kinematics mapping, which maps Euclidean rigid transformations into homogeneous points belonging to the Study quadric $\mathcal{M}_6$ in the projective space $\mathbb{P}^7$ as is expressed in

$$\mathfrak{G} = \left\{ \begin{matrix} SE(3) \to \mathbb{P}^7 \\ \alpha \to \mathbf{X} = (x_0, \ldots, x_7) \end{matrix} \right\}, \tag{9.14}$$

where $SE(3)$ is the special Lie group for Euclidean rigid transformations in three dimensions, α is any Euclidean rigid transformation, and $\mathbf{X}$ is a vector containing homogeneous coordinates which represent a point in $\mathbb{P}^7$.

9.2.1 Interpolation Algorithm

The Study quadric interpolation algorithm specifies that given a set $\mathbf{X}$ that contains three homogenous points $[\mathbf{X}_0, \mathbf{X}_1, \mathbf{X}_2]$, an interpolation curve $\mathcal{X}$ created by interpolation homogeneous points $\left[\mathbf{X}'_0, \ldots, \mathbf{X}'_N\right]$ that satisfied this set can be formulated as follows:

$$\mathcal{X} = f_0(t)\mathbf{X}_2^T M \mathbf{X}_3 \mathbf{X}_1 + f_1(t)\mathbf{X}_1^T M \mathbf{X}_3 \mathbf{X}_2 + f_0(t)\mathbf{X}_1^T M \mathbf{X}_2 \mathbf{X}_3, \tag{9.15}$$

where t is a interpolation value between 0 and 1, $f_0(t)$, $f_1(t)$, and $f_2(t)$ are calculated as below:

$$\begin{aligned} f_0 &= (t_0 - t_1)(t_0 - t_2)(t - t_1)(t - t_2), \\ f_1 &= (t_1 - t_0)(t_1 - t_2)(t - t_0)(t - t_2), \\ f_2 &= (t_2 - t_0)(t_2 - t_1)(t - t_0)(t - t_1). \end{aligned} \tag{9.16}$$

M is the matrix of the Study quadric bilinear form defined as

$$M = \begin{bmatrix} 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} \\ \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 \end{bmatrix}. \tag{9.17}$$

This interpolation algorithm can be extended for more points using an adapted De Casteljou algorithm presented in [7] as shown in Algorithm 2. The solution is a rational motion by means of an interpolating spline involving rational sub-spline motions, see [8].

Algorithm 2 Study quadric interpolation algorithm

function SQI(M,t)
  if $|X| = 1$ **then**
    $Xi \leftarrow X$
  else
    $f_0 \leftarrow (t_0 - t_1)(t_0 - t_2)(t - t_1)(t - t_2)$
    $f_1 \leftarrow (t_1 - t_0)(t_1 - t_2)(t - t_0)(t - t_2)$
    $f_2 \leftarrow (t_2 - t_0)(t_2 - t_1)(t - t_0)(t - t_1)$
    $Mp \leftarrow \emptyset$
    $N \leftarrow |X|$
    for $i = 1$ **to** $N - 2$ **do**
      $Xp_{[i]} \leftarrow f_0 X_{[i+1]}^T M X_{[N]} X_{[1]} + f_1 X_{[1]}^T M X_{[N]} X_{[i+1]} + f_2 X_{[1]}^T M X_{[i+1]} X_{[N]}$
    $Xi \leftarrow SQI(Xp, t)$
  return Xi

9.2.2 Motor Interpolation

In the Study quadric interpolation algorithm, homogeneous points can be mapped from dual quaternions or transformation matrices. Since motors are isomorph to dual quaternions, a motor can be converted into a homogeneous point. In fact, this motor represents a vector $\overrightarrow{M} = \{M_0, \ldots, M_7\} \in \mathbb{R}^8$ which lies on the Study manifold, see Table 9.1. For this reason, we propose a modification of the interpolation algorithm, where homogeneous points are substituted by motors in order to avoid the conversion, and using geometric algebra operations instead of matrices

Table 9.1 Study quadric homogeneous point, dual quaternion, $G^{+}_{3,0,1}$ and $G_{4,1}$ motor equivalence

Homogeneous point	Dual quaternion	$G^{+}_{3,0,1}$	$G_{4,1}$
x_0	α	α	α
x_1	i	e_{23}	e_{23}
x_2	j	e_{31}	e_{31}
x_3	k	e_{12}	e_{12}
x_4	ϵ	e_{1234}	$e_{123\infty}$
x_5	ϵi	e_{41}	$e_{\infty 1}$
x_6	ϵj	e_{42}	$e_{\infty 2}$
x_7	ϵk	e_{43}	$e_{\infty 3}$

operations. Via geometric algebra, (9.15) is rewritten for motors interpolation, with three control motors as

$$M'(t) = f_0(t)C_0M_0 + f_1(t)C_1M_1 + f_2(t)C_2M_2, \tag{9.18}$$

where C_0, C_1, and C_2 value are the geometric algebra modifications of the matrix multiplication $\mathbf{X_1^T M X_2}$, $\mathbf{X_0^T M X_2}$, and $\mathbf{X_0^T M X_1}$ present in (9.15), respectively; this value is calculated via (9.19)

$$\begin{aligned} C_0 &= \frac{coeff\left(\left\langle M_1\widetilde{M}_2\right\rangle_4\right)}{2}, \\ C_1 &= \frac{coeff\left(\left\langle M_0\widetilde{M}_2\right\rangle_4\right)}{2},. \\ C_2 &= \frac{coeff\left(\left\langle M_0\widetilde{M}_1\right\rangle_4\right)}{2}. \end{aligned} \tag{9.19}$$

In CGA, (9.19) can be expressed as

$$\begin{aligned} C_0 &= \frac{I_e \cdot \left(M_1\widetilde{M}_2\right) \cdot e_0}{2}, \\ C_1 &= \frac{I_e \cdot \left(M_0\widetilde{M}_2\right) \cdot e_0}{2},. \\ C_2 &= \frac{I_e \cdot \left(M_0\widetilde{M}_1\right) \cdot e_0}{2}. \end{aligned} \tag{9.20}$$

Similar to Algorithm 2, for more points, (9.18) can be extended using the De Casteljou algorithm as shown in Algorithm 3. The interpolation of motors can be used to

Algorithm 3 Motor Interpolation on CGA

function CGAMI(M,t)
  if $|M| = 1$ **then**
    $Mi \leftarrow M$
  else
    $f_0 \leftarrow (t_0 - t_1)(t_0 - t_2)(t - t_1)(t - t_2)$
    $f_1 \leftarrow (t_1 - t_0)(t_1 - t_2)(t - t_0)(t - t_2)$
    $f_2 \leftarrow (t_2 - t_0)(t_2 - t_1)(t - t_0)(t - t_1)$
    $Mp \leftarrow \emptyset$
    $N \leftarrow |M|$
    for $i = 1$ **to** $N - 2$ **do**
      $C_0 \leftarrow 0.5 coeff(\langle M_{[i+1]} \widetilde{M_{[N]}} \rangle_4)$
      $C_1 \leftarrow 0.5 coeff(\langle M_{[1]} \widetilde{M_{[N]}} \rangle_4)$
      $C_2 \leftarrow 0.5 coeff(\langle M_{[1]} \widetilde{M_{[i+1]}} \rangle_4)$
      $Mp_{[i]} \leftarrow f_0 C_0 M_{[1]} + f_1 C_1 M_{[i+1]} + f_2 C_2 M_{[N]}$
    $Mi \leftarrow CGAMI(Mp, t)$
  return Mi

calculate the new position of geometric entities as follows:

$$O'(t) = M'(t) O(t) \widetilde{M'(t)}, \tag{9.21}$$

where $O(t)$ can be any geometric entity of point, line, plane, circle, sphere, or pair of points expressed in conformal geometric algebra [9].

Figure 9.1 shows the interpolation of these geometric objects under the action of a motor $M'(t)$.

9.3 GPU Acceleration

Given that the computational complexity in an n-dimensional geometric algebra is $O(2^{2N})$ for binary operations and $O(2^N)$ for unary operations, and the motor interpolation algorithm for M interpolated motors using K motors has a computational complexity $O(MK)$; for a real-time implementation of the interpolation algorithm, an arithmetic accelerator is required. The most commonly used arithmetic accelerators are based on FPGA, ASIC, and GPU.

The proposed accelerator is based on CPU–GPU co-design; the selected GPU is a GTX-970 by Nvidia with compute capability 5.2 but can be implemented on any GPU with CUDA support. In our implementation approach, the algorithm was developed to calculate a single or multiple interpolation curves in parallel. For a single interpolation curve, the default GPU was implemented, where each motor

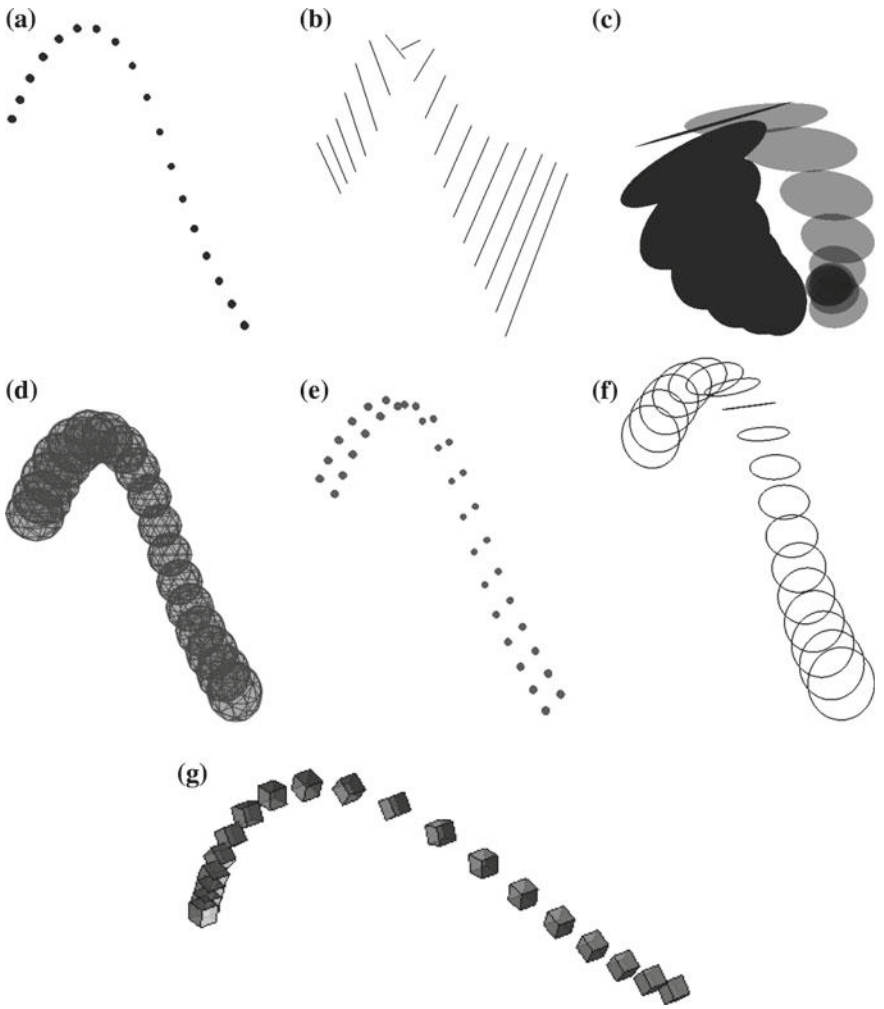


Fig. 9.1 Interpolation of **a** points, **b** lines, **c** planes, **d** spheres, **e** pair of points, and **f** circles using CGA motor interpolation curve (**g**)

is calculated in one thread launched in a stream. For multiple curves, the multi-streaming implementation was selected, in order to overlap the streams.

9.3.1 Methodology

The Compute Unified Device Architecture (CUDA) is a parallel computing platform developed by Nvidia for highly parallel implementations using its GPUs. CUDA is based on C/C++, but other languages like Python or Fortran can be used. The

proposed accelerator was written with CUDA C++ for GPU acceleration and C++ for CPU interface.

Due to GTX-970 GPU does not support recursive algorithm, the Algorithm 3 is translated into an equivalent iterative version resulting in Algorithm 4. This algorithm can be implemented for one curve interpolation (using one stream) or multiple curves (via multi-streaming) using the GPU global memory which is the largest memory in GPU, has the highest latency, and can be accessed on the device from any stream in multi-stream applications. This is useful, when the number of motors (of the motor algebra) to be calculated is large, or when we want to calculate multiple curves or launch several algorithms using the same GPU via multi-streaming.

9.3.2 Standard GPU Implementation

The main difference between GPU and CPU is the architecture: CPU has a reduced set of cores, designed for sequential applications. On the other hand, GPU has a large number of cores for highly parallel implementations. CPU and GPU architectures are shown in Fig. 9.2. GPU cores are designed to run asynchronously using threads; each core has a limit number of threads (32, 64, 128, 256, 512). For GPU implementation, when a kernel (parallel implementation in GPU) is launched, the number of threads and cores is required, and the number of cores is calculated in function of the selected threads per block. In our implementation, each thread launches a copy of the algorithm for calculating an interpolated motor.

9.3.3 Multi-streaming Implementation

When a kernel is launched in GPU, runs explicitly or implicitly on a stream, a stream refers to a sequence of asynchronous operations that execute on a device in the order issued by the host code. A stream encapsulates these operations, maintains

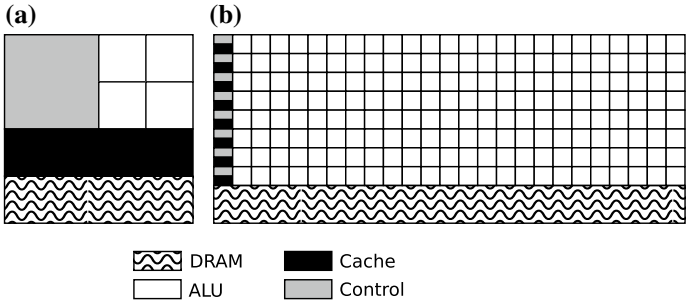


Fig. 9.2 Architecture comparison of **a** CPU and **b** GPU

Algorithm 4 Motor Interpolation algorithm on CGA (iterative)

function CGAMI(M,t)
  if $|M| = 1$ **then**
    $Mi \leftarrow M$
  else
    $f_0 \leftarrow (t_0 - t_1)(t_0 - t_2)(t - t_1)(t - t_2)$
    $f_1 \leftarrow (t_1 - t_0)(t_1 - t_2)(t - t_0)(t - t_2)$
    $f_2 \leftarrow (t_2 - t_0)(t_2 - t1)(t - t_0)(t - t_1)$
    $Mp \leftarrow \emptyset$
    $i \leftarrow |M|$
    while $i > 0$ **do**
      for $j = 1$ **to** $i - 2$ **do**
        $C_0 \leftarrow 0.5coeff(\langle M_{[i+1]}\widetilde{M_{[N]}}\rangle_4)$
        $C_1 \leftarrow 0.5coeff(\langle M_{[1]}\widetilde{M_{[N]}}\rangle_4)$
        $C_2 \leftarrow 0.5coeff(\langle M_{[1]}\widetilde{M_{[i+1]}}\rangle_4)$
        $Mp_{[j]} \leftarrow f_0C_0M_{[1]} + f_1C_1M_{[i+1]} + f_2C_2M_{[N]}$
      $M \leftarrow Mp$
      $i \leftarrow i - 2$
    $Mi \leftarrow Mp_{[1]}$
  return Mi

their ordering, permits operations to be queued in the stream to be executed after all preceding operations, and allows for querying the status of queued operations. These operations can include data transfer and kernel launches. Modern GPUs allow a parallel technique called multi-streaming, where multiple streams are launched, the transference data and kernel launch can be overlapped, and Fig. 9.3 shows this technique.

For multiple curves' calculation in parallel, we propose the use of multi-streaming implementation, where each curve is calculated in an independent stream.

9.4 Experimental Results

In this section, we compare the Study quadric and the motor interpolation in order to know which algorithm offers the best speed performance. For the comparison, several motors were calculated (20, 50, 100, 200, 500, 1000, 2000, 5000, 10,000, and 20,000 points) using 15 test groups of 5 motors. The experiments were implemented using MATLAB on a Desktop with Intel core I7 microprocessor at 2.4 GHz, 32 GB of RAM, and a GTX-970 GPU by Nvidia.

The result of the experiment is summarized in Fig. 9.4. The time execution in both algorithms grew Linearly, while the number of points increases (the algorithm complexity is $O(n)$), but motor interpolation version is faster than the Study quadric interpolation (for 20,000 points, it is 28x times faster).

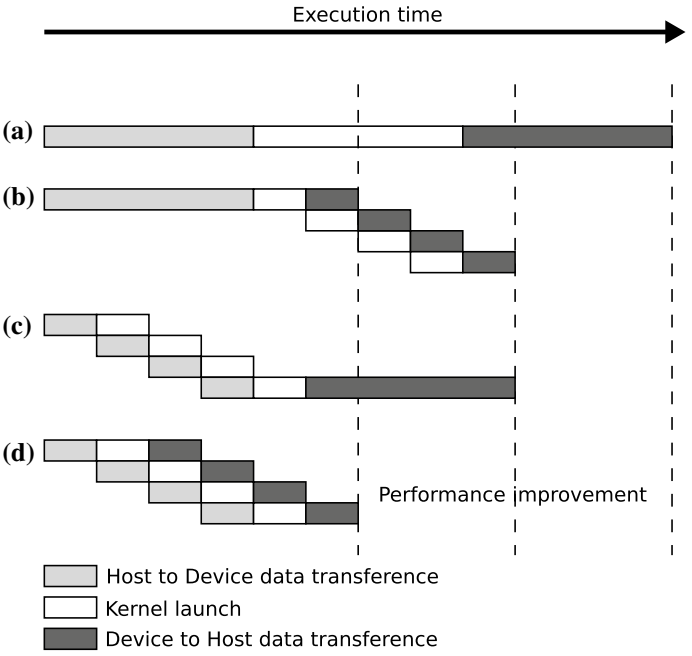


Fig. 9.3 Concurrent implementation using streams. **a** one-stream version, **b** concurrent version for device-to-host asynchronous memory copy, **c** concurrent version for host-to-device asynchronous memory copy, **d** concurrent version for host-to-device and device-to-host asynchronous memory copy

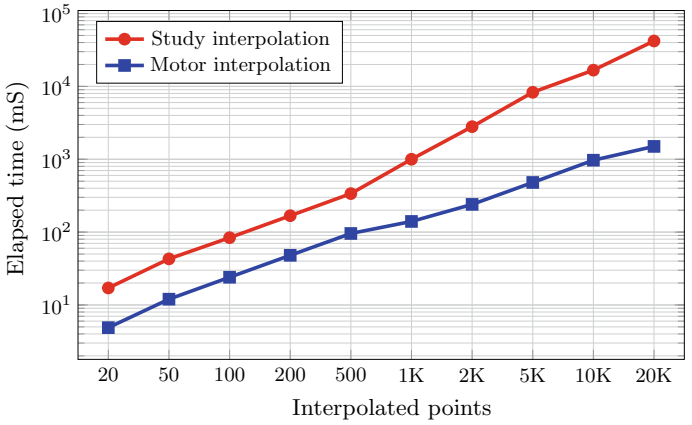


Fig. 9.4 Study quadric and motor interpolation speed-up analysis

9.4.1 Serial and Parallel Speed-Up Comparison

Due to the results shown in Fig. 9.4, the motor interpolation algorithm was selected for sequential and parallel implementation using the same hardware and parameters of the algorithm comparison.

For sequential implementation, C++ language was selected and CUDA C++ for parallel version using 32 threads per block in the kernel. The interpolated motors were calculated with single- and double-precision floating-point representation. All the measures include only the algorithm execution, excluding data transference, file writing, and reading. All the measures include only the algorithm execution, excluding data transference, file writing, and reading. The results obtained from the experiment are shown in Fig. 9.5.

As it was expected, GPU has the best time performance for calculating the interpolation algorithm. While the time execution in CPU increases linearly, the time in GPU for 500 or fewer points are almost constant. For 20,000 motors, GPU implementation runs 79x and 60x times faster than CPU for single and double floating points, respectively.

9.4.2 Multi-streaming Implementation

For multi-streaming implementation, 16 different interpolation curves were calculated using the concurrent version for host-to-device and device-to-host asynchronous memory copy with four configurations (1, 2, 4, and 8 streams) with 32 threads per block. Measure times, including data transference, are shown in Fig. 9.6.

By observing the figure, using multi-streaming reduce the execution time considerably; in this case, for 20,000 points, 8 streams version is 2.7x, 1.9x, and 1.6x times faster than 1, 2, and 4 streams, respectively.

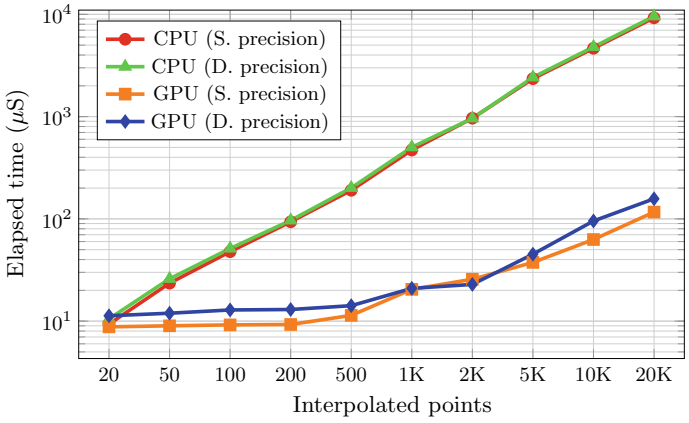


Fig. 9.5 CPU and GPU speed-up analysis

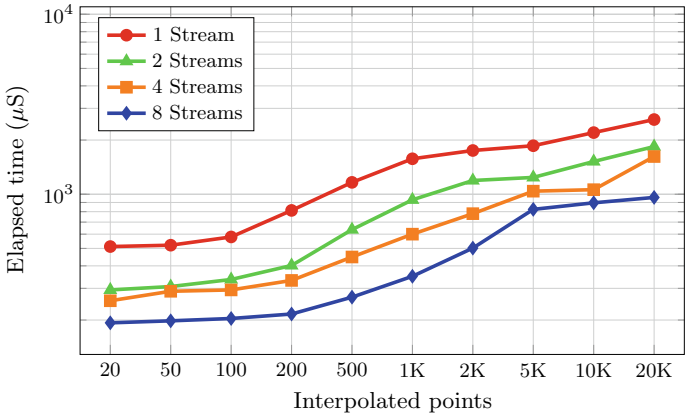


Fig. 9.6 Multi-stream speed-up analysis

9.5 Conclusions

In this chapter, an optimized version of the Study quadric interpolation algorithm using conformal geometric algebra was presented. A MATLAB implementation shows that conformal geometric algebra interpolation is 28x times faster than the Study quadric interpolation for 20,000 motors.

Also, CPU and GPU versions of motor interpolation algorithm were compared, for single and double floating-point precision. Experimental results show that GPU is 79x and 60 times faster than the CPU implementation for single and double precisions, respectively.

Finally, 16 interpolation curves were calculated via multi-streaming; the obtained results show that multi-streaming reduces the execution time for multiple curves' interpolation.

In the future, we aim to implement the algorithm using multiple GPU and improve the algorithm to reduce the data dependency. We also aim to create an OpenCL library of the algorithm and use another language, like Python.

References

1. Ureña-Ponce, O., Soria-García, G., Martínez-Terán, G., Ortega-Cisneros, S., & Bayro-Corrochano, E. (2019). Accelerating interpolation in conformal geometric algebra using GPU. *Journal of Advances on Applications of Clifford Algebra*.
2. Bayro-Corrochano, E., Garza-Burgos, A. M., & Del Valle-Padilla, J. L. (2019). Geometric intuitive techniques for human machine interaction in medical robotics. *Journal of Social Robotics*.
3. Etzel, K. R., & McCarthy, J. M. (1996). Spatial motion interpolation in an image space of so(4). In *Proceedings of The 1996 ASME Design Engineering Technical Conference and Computers in Engineering Conference*.

4. Kavan, L., Collins, S., Zara, J., & O'Sullivan, C. (2006). Geometric skinning with approximate dual quaternion blending. Technical Report TCD-CS-2006-46, The University of Dublin, Trinity College.
5. Jüttler, B. (1994). Visualization of moving objects using the dual quaternion curves. *Computer and Graphics, 18*(3), 315–326.
6. Gfrerrer, A. (2010). Study's kinematic mapping—A tool for motion design. In *Advances in robot kinematics* (pp. 7–16). Springer.
7. Klawitter, D., (2010). Documentation-Kinematic Toolbox. Technische Universität Dresden.
8. Prautzsch, H., Boehm, W., & Paluszny, M. (2013). *Bézier and B-spline techniques*. Springer Science & Business Media.
9. Bayro-Corrochano, E. (2010). *Geometric computing for wavelet transforms, robot vision, learning, control and action*. London: Springer-Verlag.

Chapter 10
Robot Kinematics

Mechanics is the paradise of the mathematical sciences because by means of it one comes to the fruits of mathematics.
Leonardo Da Vinci (April 15, 1452–May 2, 1519)

10.1 Introduction

This chapter presents the formulation of robot manipulator kinematics within the geometric algebra framework. In this algebraic system, the 3D Euclidean motion of points, lines, and planes can be advantageously represented using the algebra of motors. The computational complexity of direct and indirect kinematics and other problems concerning robot manipulators are dependent on the robot's degrees of freedom as well on its geometric characteristics. Our approach makes possible a direct algebraic formulation of the concrete problem in such a way that it reflects the underlying geometric structure. This is achieved by describing parts of the problem based on motor representations of points, lines, planes, circles, and spheres where necessary. The chapter presents the formulation and computation of closed-form solutions of direct and indirect kinematics for standard robot manipulators and a simple example of a grasping task. The flexible method presented here widens the current standard point or line representation-based approaches for the treatment of problems related to robot manipulators. We present the computation of the inverse kinematics in the motor algebra using points, lines, and planes, and as an extension we solve the same problem but using conformal geometric algebra which allows us for the representation and also the use of circles and spheres.

© Springer Nature Switzerland AG 2020

E. Bayro-Corrochano, *Geometric Algebra Applications Vol. II*,
https://doi.org/10.1007/978-3-030-34978-3_10

10.2 Screw Theory

In this section, we give some definitions and concepts using screw lines useful for robotics.

10.2.1 Reciprocal Screws

The reciprocal product of two screws, say $\boldsymbol{L}_1 = \boldsymbol{n}_1 + e_\infty \boldsymbol{m}_1$ and $\boldsymbol{L}_2 = \boldsymbol{n}_2 + e_\infty \boldsymbol{m}_2$, is defined by

$$\boldsymbol{L}_1 \circ \boldsymbol{L}_2 = \boldsymbol{n}_1 \cdot \boldsymbol{m}_2 + \boldsymbol{n}_2 \cdot \boldsymbol{m}_1, \tag{10.1}$$

where the symbol $\circ$ denotes the reciprocal product of two screws lines. When the origin of the coordinate system shifts from the origin to point $\boldsymbol{p}$, the two line screws $\boldsymbol{L}_1$ and $\boldsymbol{L}_2$ become

$$\boldsymbol{L}_1' = \boldsymbol{n}_1 + e_\infty(\boldsymbol{m}_1 + \boldsymbol{p} \wedge \boldsymbol{n}_1) \quad \boldsymbol{L}_2' = \boldsymbol{n}_2 + e_\infty(\boldsymbol{m}_2 + \boldsymbol{p} \wedge \boldsymbol{n}_2). \tag{10.2}$$

The reciprocal product of the two new screw lines is

$$\boldsymbol{L}_1' \circ \boldsymbol{L}_2' = \boldsymbol{n}_1 \cdot (\boldsymbol{m}_2 + \boldsymbol{p} \wedge \boldsymbol{n}_2) + \boldsymbol{n}_2 \cdot (\boldsymbol{m}_1 + \boldsymbol{p} \wedge \boldsymbol{n}_1) = \boldsymbol{L}_1 \circ \boldsymbol{L}_2. \tag{10.3}$$

This result indicates that the reciprocal product of two screws is not origin dependent. The reciprocal product of two screws represents the work produced by a wrench acting on a rigid body undergoing an infinitesimal twist.

10.3 Elementary Transformations of Robot Manipulators

The study of the rigid motion of objects in 3D space plays an important role in robotics. In order to linearize the rigid motion of the Euclidean space, homogeneous coordinates are normally utilized. That is why in the geometric algebra framework, we choose the special or degenerated geometric algebra to extend the algebraic system from 3D Euclidean space to 4D space. In this system, we can nicely model the motion of points, lines, and planes with computational advantages and geometric insight. Let us start with a description of the basic elements of robot manipulators in terms of the special or degenerated geometric algebra $\mathcal{G}^+_{3,0,1}$ or motor algebra. The most basic parts of a robot manipulator are revolute joints, prismatic joints, connecting links, and the end-effectors. In the following subsections, we will treat the kinematics of the prismatic and revolute manipulator parts using the 4D geometric algebra $\mathcal{G}^+_{3,0,1}$ and will illustrate an end-effector grasping task.

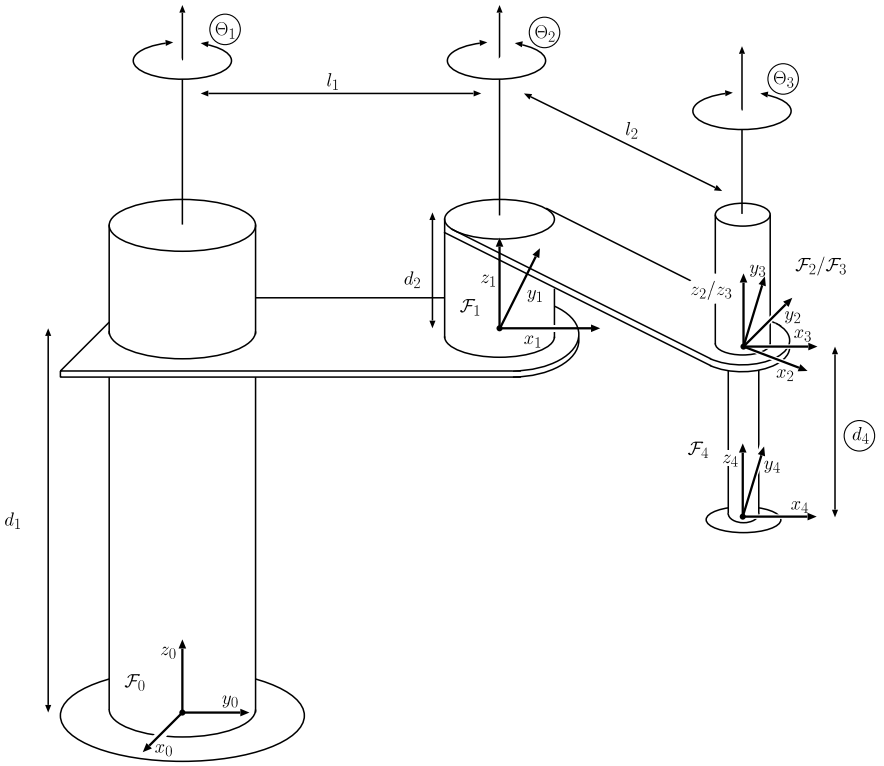


Fig. 10.1 SCARA-type manipulator according to DH parameters given in Table 10.1 (variable parameters are circled)

10.3.1 The Denavit–Hartenberg Parameterization

The computation of direct or inverse kinematics requires both an exact description of the robot manipulator's structure and its configuration. The descriptive approach used most often is known as the Denavit–Hartenberg (DH) procedure [1]. It is based on a uniform description of the position of the reference coordinate system of a particular joint relative to the next joint in consideration. Figure 10.3a shows how coordinate frames are attached to a joint of a robot manipulator. Table 10.1 presents the specifications of two robot manipulators, the SCARA and the Stanford manipulators, as shown in Figs. 10.1 and 10.2, respectively.

This tells us whether the joint is for rotation (revolute) or for translation (prismatic). The transformation of the reference coordinate system between two joints will be called a joint transition. Figure 10.3b shows the involved screws in a joint transition according to the Denavit–Hartenberg parameters. The frame, or reference coordinate system, related to the i-th joint is attached at the end of this link, and it is notated $\mathcal{F}_i$.

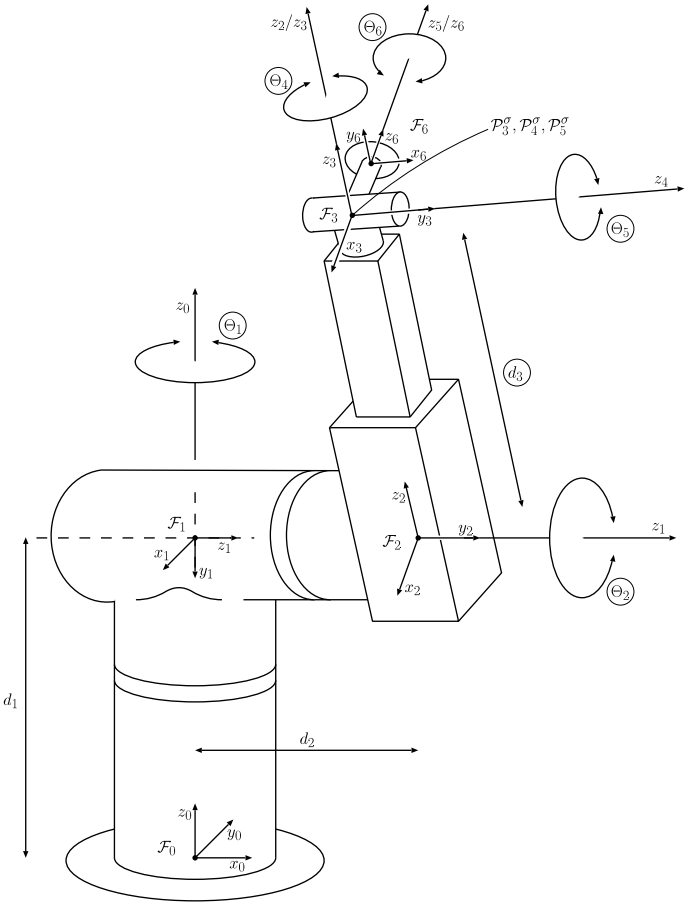


Fig. 10.2 Stanford-type manipulator according to DH parameters in Table 10.1 (variable parameters are encircled)

In Table 10.1, the parameter "v" indicates that the involved variable of the joint is variable, and the parameter "c" indicates that it is constant.

The position and orientation of the end-effector in relation to the reference coordinate system of the robot basis can be computed by linking all joint transitions. In this way, we are able to get straightforwardly the *direct kinematics*.

Conversely, in the case of the inverse kinematics, given the position and orientation of the end-effector, we must find values of the variable parameters of the joint transitions which satisfy this requirement. In the following sections, we will provide more details for the computation of the direct and inverse kinematics of robot manipulators.

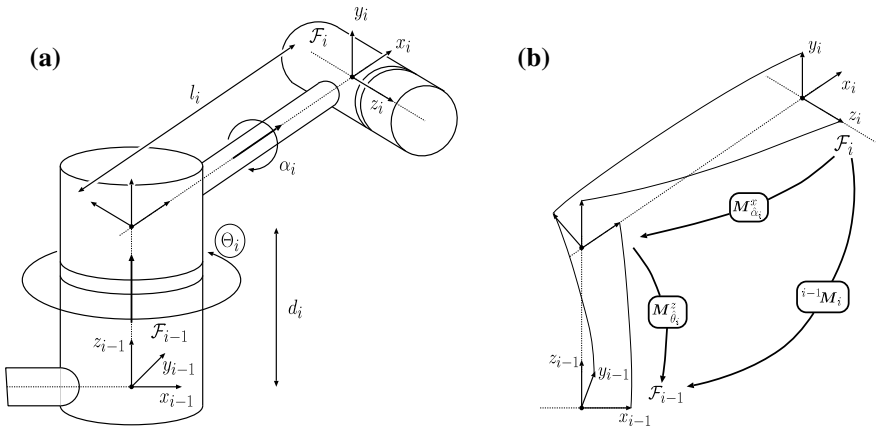


Fig. 10.3 Motor description of a manipulator joint: **a** i-th joint of a robot manipulator and attached coordinate frames according to Denavit–Hartenberg; the encircled θ_i is the variable parameter; **b** transformation from frame $\mathcal{F}_i$ to $\mathcal{F}_{i-1}$ (motor ${}^{i-1}\boldsymbol{M}_i$ consists of two screw transformations, $\boldsymbol{M}^x_{\hat{\alpha}_i}$ and $\boldsymbol{M}^z_{\hat{\theta}_i}$)

Table 10.1 Kinematic configuration of two robotic manipulators

Robot type	Link	Revolute θ_i	v/c	Prismatic d_i	v/c	Twist angle α_i	Link length l_i
SCARA	1	θ_1	v	d_1	c	0	l_1
	2	θ_2	v	d_2	c	0	l_2
	3	θ_3	v	0		0	0
	4	0		d_4	v	0	0
Stanford	1	θ_1	v	d_1	c	$-90°$	0
	2	θ_2	v	d_2	c	$90°$	0
	3	0		d_3	v	0	0
	4	θ_4	v	0		$-90°$	0
	5	θ_5	v	0		$90°$	0
	6	θ_6	v	d_6	c	0	0

10.3.2 Representations of Prismatic and Revolute Transformations

The transformation of any point, line, or plane between coordinate systems $\mathcal{F}_{i-1}$ and $\mathcal{F}_i$ is revolute when the involved degree of freedom is only a variable angle θ_i, and it is prismatic when the degree of freedom is only a variable length d_i. The transformation motor ${}^{i-1}\boldsymbol{M}_i$ between $\mathcal{F}_i$ and $\mathcal{F}_{i-1}$ consists of a sequence of two screw transformations, one fixed (i.e., $\boldsymbol{M}^x_{\hat{\alpha}_i}$) and another variable (i.e., $\boldsymbol{M}^z_{\hat{\theta}_i}$) (see Fig. 10.3b). Note that we use dual angles

$$\hat{\theta}_i = \theta_i + Id_i \tag{10.4}$$

$$\hat{\alpha}_i = \alpha_i + Il_i, \tag{10.5}$$

see [2]. In the revolute case, the latter equation uses as a variable parameter the angle θ_i and in the prismatic case the displacement d_i is the variable parameter. The transformation reads

$$\begin{aligned} {}^{i-1}\boldsymbol{M}_i &= \boldsymbol{M}^z_{\hat{\theta}_i}\boldsymbol{M}^x_{\hat{\alpha}_i} = \boldsymbol{T}^z_{d_i}\boldsymbol{R}^z_{\theta_i}\boldsymbol{T}^x_{l_i}\boldsymbol{R}^x_{\alpha_i}, \\ &= \left(1 + \frac{I}{2}\begin{pmatrix}0\\0\\d_i\end{pmatrix}\right)\boldsymbol{R}^z_{\theta_i}\left(1 + \frac{I}{2}\begin{pmatrix}l_i\\0\\0\end{pmatrix}\right)\boldsymbol{R}^x_{\alpha_i}. \end{aligned} \tag{10.6}$$

For the sake of clarity, the dual bivectors of the translators are given as a column vector, which makes the variable parameters explicit.

Since ${}^{i-1}\boldsymbol{M}_i\ {}^{i-1}\widetilde{\boldsymbol{M}}_i = 1$, we obtain

$${}^{i}\boldsymbol{M}_{i-1} = \widetilde{\boldsymbol{M}}^x_{\hat{\alpha}_i}\widetilde{\boldsymbol{M}}^z_{\hat{\theta}_i} = \widetilde{\boldsymbol{T}}^x_{l_i}\widetilde{\boldsymbol{R}}^x_{\alpha_i}\widetilde{\boldsymbol{T}}^z_{d_i}\widetilde{\boldsymbol{R}}^z_{\theta_i}. \tag{10.7}$$

Be aware that for the rest of the chapter ${}^{j}\boldsymbol{M}_i$ will denote a motor transformation from $\mathcal{F}_i$ to $\mathcal{F}_j$.

We will now give general expressions for the transformation of points, lines, and planes using a single parameter, θ_i or d_i, respectively, as the variable and two fixed parameters, α_i and l_i. For the joint depicted in Fig. 10.3b, a revolute transformation will take place only when θ_i varies and a prismatic transformation only when d_i varies. Now, taking a point $\boldsymbol{X}$ represented in the frame $\mathcal{F}_{i-1}$, we can describe its transformation from $\mathcal{F}_{i-1}$ to $\mathcal{F}_i$ using the motor algebra [2], with either θ_i or d_i as the variable parameter. We will call this transformation a *forward transformation*.

The multivector representation of point $\boldsymbol{X}$ related to the frame $\mathcal{F}_i$ will be expressed as ${}^{i}\boldsymbol{X}$, with

$$\begin{aligned} {}^{i}\boldsymbol{X} &= {}^{i}\boldsymbol{M}_{i-1}\ {}^{i-1}\boldsymbol{X}\ {}^{i}\widetilde{\overline{\boldsymbol{M}}}_{i-1} = \widetilde{\boldsymbol{M}}^x_{\hat{\alpha}_i}\widetilde{\boldsymbol{M}}^z_{\hat{\theta}_i}\ {}^{i-1}\boldsymbol{X}\ \overline{\boldsymbol{M}}^z_{\hat{\theta}_i}\overline{\boldsymbol{M}}^x_{\hat{\alpha}_i}, \\ &= \widetilde{\boldsymbol{T}}^x_{l_i}\widetilde{\boldsymbol{R}}^x_{\alpha_i}\widetilde{\boldsymbol{T}}^z_{d_i}\widetilde{\boldsymbol{R}}^z_{\theta_i}\ {}^{i-1}\boldsymbol{X}\ \boldsymbol{R}^z_{\theta_i}\widetilde{\boldsymbol{T}}^z_{d_i}\boldsymbol{R}^x_{\alpha_i}\widetilde{\boldsymbol{T}}^x_{l_i}, \\ &= 1 + I\,{}^{i}\boldsymbol{x}, \end{aligned} \tag{10.8}$$

where ${}^{i}\boldsymbol{x}$ is a bivector representing the 3D position of $\boldsymbol{X}$ referred to $\mathcal{F}_i$. A transformation in the reverse sense will be called a *backward transformation*. In this case, a point $\boldsymbol{X}$ referenced in the frame $\mathcal{F}_i$ is transformed to the frame $\mathcal{F}_{i-1}$, as follows:

$$\begin{aligned} {}^{i-1}\boldsymbol{X} &= {}^{i-1}\boldsymbol{M}_i\ {}^{i}\boldsymbol{X}\ {}^{i-1}\widetilde{\overline{\boldsymbol{M}}}_i = \boldsymbol{M}^z_{\hat{\theta}_i}\boldsymbol{M}^x_{\hat{\alpha}_i}\ {}^{i}\boldsymbol{X}\ \widetilde{\overline{\boldsymbol{M}}}^x_{\hat{\alpha}_i}\widetilde{\overline{\boldsymbol{M}}}^z_{\hat{\theta}_i}, \\ &= 1 + I\,{}^{i-1}\boldsymbol{x}. \end{aligned} \tag{10.9}$$

Note that the motor applied from the right side is not purely conjugated as is the case with the line. The same will be also true for a plane [2].

Consider a line $\boldsymbol{L}$ represented in the frame $\mathcal{F}_{i-1}$ by ${}^{i-1}\boldsymbol{L} = {}^{i-1}\boldsymbol{n} + I\,{}^{i-1}\boldsymbol{m}$, where $\boldsymbol{n}$ and $\boldsymbol{m}$ are bivectors indicating the orientation and moment of the line, respectively. We can write the forward transformation of this line related to the frame $\mathcal{F}_i$ according to [2]

$$\begin{aligned} {}^{i}\boldsymbol{L} &= {}^{i}\boldsymbol{M}_{i-1}\ {}^{i-1}\boldsymbol{L}\ {}^{i}\widetilde{\boldsymbol{M}}_{i-1} = \widetilde{\boldsymbol{M}}^{x}_{\hat{\alpha}_i}\widetilde{\boldsymbol{M}}^{z}_{\hat{\theta}_i}\ {}^{i-1}\boldsymbol{L}\ \boldsymbol{M}^{z}_{\hat{\theta}_i}\boldsymbol{M}^{x}_{\hat{\alpha}_i} \\ &= {}^{i}\boldsymbol{n} + I\,{}^{i}\boldsymbol{m}. \end{aligned} \tag{10.10}$$

Its backward transformation reads

$$\begin{aligned} {}^{i-1}\boldsymbol{L} &= {}^{i-1}\boldsymbol{M}_{i}\ {}^{i}\boldsymbol{L}\ {}^{i-1}\widetilde{\boldsymbol{M}}_{i} = \boldsymbol{M}^{z}_{\hat{\theta}_i}\boldsymbol{M}^{x}_{\hat{\alpha}_i}\ {}^{i}\boldsymbol{L}\ \widetilde{\boldsymbol{M}}^{x}_{\hat{\alpha}_i}\widetilde{\boldsymbol{M}}^{z}_{\hat{\theta}_i} \\ &= {}^{i-1}\boldsymbol{n} + I\,{}^{i-1}\boldsymbol{m}. \end{aligned} \tag{10.11}$$

Finally, the forward transformation of a plane $\boldsymbol{H}$ represented in $\mathcal{F}_{i-1}$ reads

$$\begin{aligned} {}^{i}\boldsymbol{H} &= {}^{i}\boldsymbol{M}_{i-1}\ {}^{i-1}\boldsymbol{H}\ {}^{i}\widetilde{\overline{\boldsymbol{M}}}_{i-1} = \widetilde{\boldsymbol{M}}^{x}_{\hat{\alpha}_i}\widetilde{\boldsymbol{M}}^{z}_{\hat{\theta}_i}\ {}^{i-1}\boldsymbol{H}\ \overline{\boldsymbol{M}}^{z}_{\hat{\theta}_i}\overline{\boldsymbol{M}}^{x}_{\hat{\alpha}_i} \\ &= {}^{i}\boldsymbol{n} + I\,{}^{i}d_H; \end{aligned} \tag{10.12}$$

and, as above, its backward transformation equation is

$$\begin{aligned} {}^{i-1}\boldsymbol{H} &= {}^{i-1}\boldsymbol{M}_{i}\ {}^{i}\boldsymbol{H}\ {}^{i-1}\widetilde{\overline{\boldsymbol{M}}}_{i} = \boldsymbol{M}^{z}_{\hat{\theta}_i}\boldsymbol{M}^{x}_{\hat{\alpha}_i}\ {}^{i}\boldsymbol{H}\ \widetilde{\overline{\boldsymbol{M}}}^{x}_{\hat{\alpha}_i}\widetilde{\overline{\boldsymbol{M}}}^{z}_{\hat{\theta}_i} \\ &= {}^{i-1}\boldsymbol{n} + I\,{}^{i-1}d_H. \end{aligned} \tag{10.13}$$

10.3.3 Grasping by Using Constraint Equations

In this subsection, we will illustrate grasping as a manipulation-related task *grasping operation* that involves the positioning of a two-finger grasper in front of a static object. Figure 10.4 shows the grasper and the considered object $\boldsymbol{O}$. The manipulator must move the grasper near to the object, and together they must fulfill some conditions in order to grasp the object firmly. To determine the overall transformation ${}^{0}\boldsymbol{M}_n$, which moves the grasper into an appropriate grasping position, we claim that ${}^{0}\boldsymbol{M}_n$ must fulfill the three constraints below. In order to formulate these constraints, we can take advantages of the point, line, and plane representations of the motor algebra. In the following explanation, we assume that the representations of geometric entities attached to the object $\boldsymbol{O}$ in frame $\mathcal{F}_0$ are known.

Attitude condition: The grasping movement of the two fingers should be in the reference plane $\boldsymbol{H}_O$ of $\boldsymbol{O}$—that is, the yz-plane of the end-effector frame $\mathcal{F}_n$ should

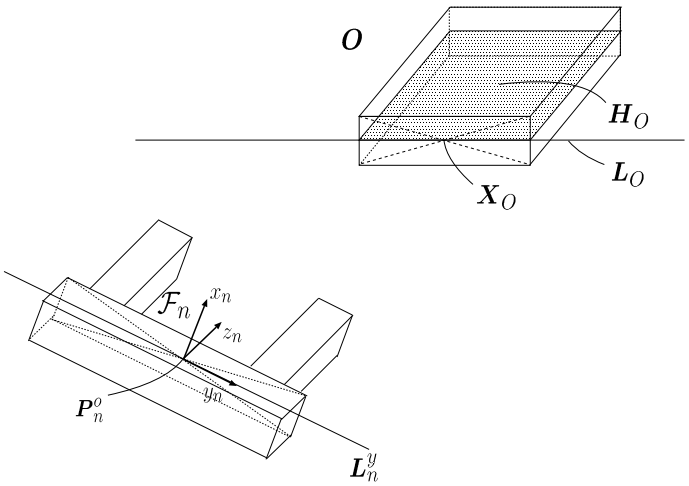


Fig. 10.4 Two-finger grasper approaching an object

be equal to the reference plane $\boldsymbol{H}_O$. The *attitude condition* can be simply formulated in terms of a plane equation, as follows:

$$ {}^0\boldsymbol{M}_n \, {}^n\boldsymbol{H}_n^{yz} \, {}^0\widetilde{\boldsymbol{M}}_n - {}^0\boldsymbol{H}_O \approx 0, \tag{10.14} $$

where ${}^n\boldsymbol{H}_n^{yz} = (1,0,0)^T + I\ 0 = (1,0,0)^T$ (see Fig. 10.4).

Alignment condition: After the application of the motor ${}^0\boldsymbol{M}_n$, the grasper and the object should be in a parallel alignment—that is, the direction of the y-axis and the line $\boldsymbol{L}_O$ should be the same. The *alignment condition* can be simply expressed in terms of a line equation,

$$ \langle {}^0\boldsymbol{M}_n \, {}^n\boldsymbol{L}_n^y \, {}^0\widetilde{\boldsymbol{M}}_n \rangle_d - \langle {}^0\boldsymbol{L}_O \rangle_d \approx 0, \tag{10.15} $$

where ${}^n\boldsymbol{L}_n^y = (0,1,0)^T + I(0,0,0)^T = (0,1,0)^T$ and $\langle \boldsymbol{L} \rangle_d$ denotes the direction components of line $\boldsymbol{L}$.

Touching condition: The motion ${}^0\boldsymbol{M}_n$ should also guarantee that the grasper is in the correct grasping position—that is, the origin $\boldsymbol{P}_n^o$ of the end-effector frame $\mathcal{F}_n$ should touch the reference point $\boldsymbol{X}_O$ of $\boldsymbol{O}$. The formulation of this constraint in our framework is

$$ {}^0\boldsymbol{M}_n \, {}^n\boldsymbol{P}_n^o \, {}^0\widetilde{\boldsymbol{M}}_n - {}^0\boldsymbol{X}_O \approx 0. \tag{10.16} $$

With these three conditions, we can obtain constraints for the components of ${}^0\boldsymbol{M}_n$, and we can determine ${}^0\boldsymbol{M}_n$ numerically. The next step is to determine the variable joint parameters of the robot manipulator which lead to the position and orientation of the end-effector frame $\mathcal{F}_n$ described by ${}^0\boldsymbol{M}_n$. This problem is called the inverse kinematics problem of robot manipulators and will be treated in Sect. 10.5.

10.4 Direct Kinematics of Robot Manipulators

The determination of direct kinematics involves the computation of the position and orientation of the end-effector or frame $\mathcal{F}_n$ given the parameters of the joint transitions (see Fig. 10.5). In this section, we will show how the direct kinematics can be computed when we use as geometric objects points, lines, or planes. The notation we use for points, lines, and planes is shown in Fig. 10.6. The direct kinematics for the general case of a manipulator with n joints can be written as

$$ {}^0\boldsymbol{M}_n = {}^0\boldsymbol{M}_1\,{}^1\boldsymbol{M}_2\,{}^2\boldsymbol{M}_3 \cdots {}^{n-1}\boldsymbol{M}_n = \prod_{i=1}^{n} {}^{i-1}\boldsymbol{M}_i. \tag{10.17} $$

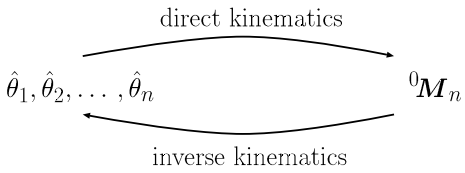


Fig. 10.5 Direct and inverse kinematics

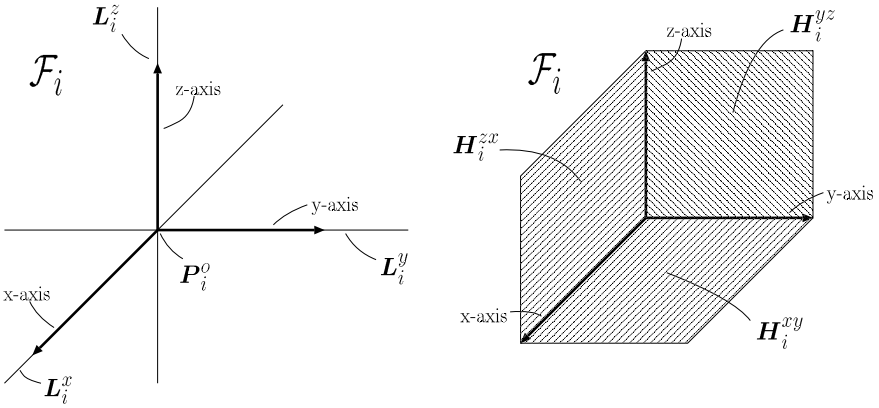


Fig. 10.6 Notations for origin, coordinate axis, and coordinate planes for frame-specific entities

We can formulate straightforwardly the direct kinematics in terms of point, line, or plane representations as follows:

$$
\begin{aligned}
{}^0\boldsymbol{X} &= {}^0\boldsymbol{M}_n \; {}^n\boldsymbol{X} \; {}^0\widetilde{\boldsymbol{M}}_n = \prod_{i=1}^{n} {}^{i-1}\boldsymbol{M}_i \; {}^n\boldsymbol{X} \; \prod_{i=1}^{n} {}^{n-i}\widetilde{\boldsymbol{M}}_{n+1-i}, \\
{}^0\boldsymbol{L} &= \prod_{i=1}^{n} {}^{i-1}\boldsymbol{M}_i \; {}^n\boldsymbol{L} \; \prod_{i=1}^{n} {}^{n-i}\widetilde{\boldsymbol{M}}_{n+1-i}, \\
{}^0\boldsymbol{H} &= \prod_{i=1}^{n} {}^{i-1}\boldsymbol{M}_i \; {}^n\boldsymbol{H} \; \prod_{i=1}^{n} {}^{n-i}\widetilde{\boldsymbol{M}}_{n+1-i}.
\end{aligned} \tag{10.18}
$$

Now, let us write the motor ${}^0\boldsymbol{M}_4$ for the direct kinematics for points, lines, and planes in the form of equation (10.18) for the SCARA manipulator specified by the Denavit–Hartenberg parameters of Table 10.1. First, using Eq. (10.17) with $n = 4$, we can easily express the required motor ${}^0\boldsymbol{M}_4$, as follows:

$$
\begin{aligned}
{}^0\boldsymbol{M}_4 &= {}^0\boldsymbol{M}_1 {}^1\boldsymbol{M}_2 {}^2\boldsymbol{M}_3 {}^3\boldsymbol{M}_4 = (\boldsymbol{M}^z_{\hat{\theta}_1} \boldsymbol{M}^x_{\hat{\alpha}_1}) \cdots (\boldsymbol{M}^z_{\hat{\theta}_4} \boldsymbol{M}^x_{\hat{\alpha}_4}), \\
&= (\boldsymbol{T}^z_{d_1} \boldsymbol{R}^z_{\theta_1} \boldsymbol{T}^x_{l_1} \boldsymbol{R}^x_{\alpha_1}) \cdots (\boldsymbol{T}^z_{d_4} \boldsymbol{R}^z_{\theta_4} \boldsymbol{T}^x_{l_4} \boldsymbol{R}^x_{\alpha_4}), \\
&= \left(1 + \frac{I}{2} \begin{pmatrix} 0 \\ 0 \\ d_1 \end{pmatrix}\right) \boldsymbol{R}^z_{\theta_1} \left(1 + \frac{I}{2} \begin{pmatrix} l_1 \\ 0 \\ 0 \end{pmatrix}\right) \left(1 + \frac{I}{2} \begin{pmatrix} 0 \\ 0 \\ d_2 \end{pmatrix}\right), \\
&\quad \boldsymbol{R}^z_{\theta_2} \left(1 + \frac{I}{2} \begin{pmatrix} l_2 \\ 0 \\ 0 \end{pmatrix}\right) \boldsymbol{R}^z_{\theta_3} \left(1 + \frac{I}{2} \begin{pmatrix} 0 \\ 0 \\ d_4 \end{pmatrix}\right).
\end{aligned} \tag{10.19}
$$

Note that translators with zero translation and rotors with zero angle become 1.

Applying the motor ${}^0\boldsymbol{M}_4$ from the left and ${}^0\widetilde{\boldsymbol{M}}_4$ from the right for point and plane equations, and the motor ${}^0\boldsymbol{M}_4$ from the left and ${}^0\tilde{\boldsymbol{M}}_4$ from the right for line equations, as indicated by Eq. (10.18), we derive the direct kinematics equations of points, lines, and planes for the SCARA robot manipulator.

10.4.1 Maple Program for Motor Algebra Computations

Since the nature of our approach requires symbolic computation, we chose the Maple computer package to develop a comfortable program for computations in the frame of different geometric algebras. To employ the program, we simply have to specify the vector basis of the selected geometric algebra. The program also employs a variety of useful algebraic operators to carry out computations involving reversion, Clifford conjugations, inner and wedge operations, rotations, translations, motors, extraction of the i-blade of a multivector, etc. Here, as a first illustration, we use the program for computations in the motor algebra framework $\mathcal{G}^+_{3,0,1}$ to compute the direct kinematic

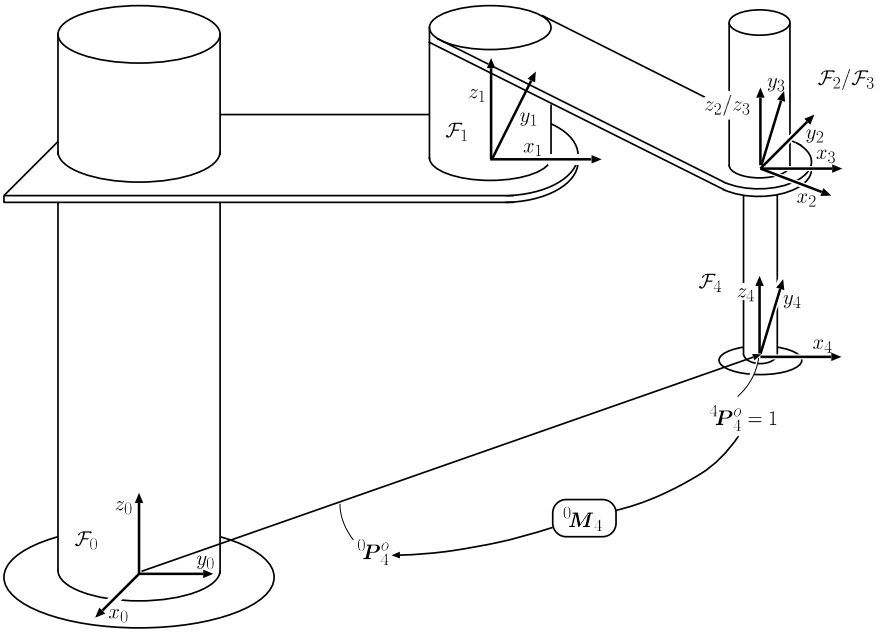


Fig. 10.7 Representation ${}^0\boldsymbol{P}_4^o$ of $\boldsymbol{P}_4^o$ in frame $\mathcal{F}_0$ is computed using ${}^0\boldsymbol{M}_4$

equation of the origin $\boldsymbol{P}_4^o$ of $\mathcal{F}_4$ for the SCARA manipulator specified by the Denavit–Hartenberg parameters of Table 10.1. Figure 10.7 shows the frames and the point $\boldsymbol{P}_4^o$ referred to $\mathcal{F}_0$. The final result of the computer calculation is

$$
\begin{aligned}
{}^0\boldsymbol{P}_4^o = {}^0\boldsymbol{M}_4\ {}^4\boldsymbol{P}_4^o\ {}^0\widetilde{\boldsymbol{M}}_4 &= {}^0\boldsymbol{M}_4 \left(1 + I \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}\right)\ {}^0\widetilde{\boldsymbol{M}}_4, \\
&= 1 + I \begin{pmatrix} l_2\cos(\theta_1+\theta_2) + l_1\cos(\theta_1) \\ l_2\sin(\theta_1+\theta_2) + l_1\sin(\theta_1) \\ d_1 + d_2 + d_4 \end{pmatrix}. \qquad (10.20)
\end{aligned}
$$

10.5 Inverse Kinematics of Robot Manipulators Using Motor Algebra

Since calculations for the inverse kinematics are more complex than those for direct kinematics, our aim should be to find a systematic approach that will exploit the point, line, and plane motor algebra representations. Unfortunately, the procedure for the computation of the inverse kinematics is not amenable to a general formulation as in the case of the direct kinematics Eq. (10.17). That is why, we are compelled to

choose a real robot manipulator and compute its inverse kinematics in order to show all the characteristics of the computational assumptions.

The Stanford robot manipulator is well known among researchers concerned with the design of strategies for the symbolic computation of the inverse kinematics. According to Table 10.1, the variable parameters to be computed are θ_1, θ_2, θ_4, θ_5, θ_6, and d_3. Using the Stanford robot manipulator, we will show that the motor algebra approach gives us the freedom to switch between the point, line, or plane representation, according to the geometrical circumstances. This is one of the most important advantages of our motor algebra approach.

Mechanically, the Stanford manipulator can be divided into two basic components. The first comprises joints 1, 2, and 3 and is dedicated to the task of general positioning. The second comprises joints 4, 5, and 6, and is dedicated to the wrist-like orientation of the end-effector. Since the philosophy of our approach relies on the application of point, line, or plane representation where it is needed, we must first evaluate, for any given case, which of these three representations are suitable for the joint transitions. In this way, a better geometric insight is guaranteed, and the solution method is more easily developed. The first three joints of the Stanford manipulator are used to position the origin of the coordinate frame $\mathcal{F}_3$. Therefore, we apply a point representation to describe this part of the problem. The last three joints are used to achieve the desired orientation of the end-effector frame. For the formulation of this subproblem, we use a line and a plane representation because these entities allow us to model orientations.

10.5.1 The Rendezvous Method

The next important step is to represent the motor transformations from the beginning of a chain of joint transitions to their completion, and vice versa, as depicted in Fig. 10.8. As a result, we gain a set of equations for each meeting point. In each of these points, the forward equation is equivalent to the backward equation. We use these equalities as a guide in the computation of the unknowns. We will call this procedure the *rendezvous method*. This simple idea has proved to be very useful as a strategy for the solution of the inverse kinematics.

10.5.2 Computing θ_1, θ_2, and d_3 Using a Point

In the case of the Stanford manipulator, the orientation and position of frame $\mathcal{F}_6$ uniquely determine the position of frame $\mathcal{F}_3$. An explanation for this relation follows.

The position of frame $\mathcal{F}_3$ with respect to $\mathcal{F}_0$ is described by the multivector representation ${}^0\boldsymbol{P}_3^o$ of $\boldsymbol{P}_3^o$ in $\mathcal{F}_0$. By successive forward transformation applied on ${}^3\boldsymbol{P}_3^o = 1$, we find the representation ${}^6\boldsymbol{P}_3^o$ of $\boldsymbol{P}_3^o$ in $\mathcal{F}_6$ by

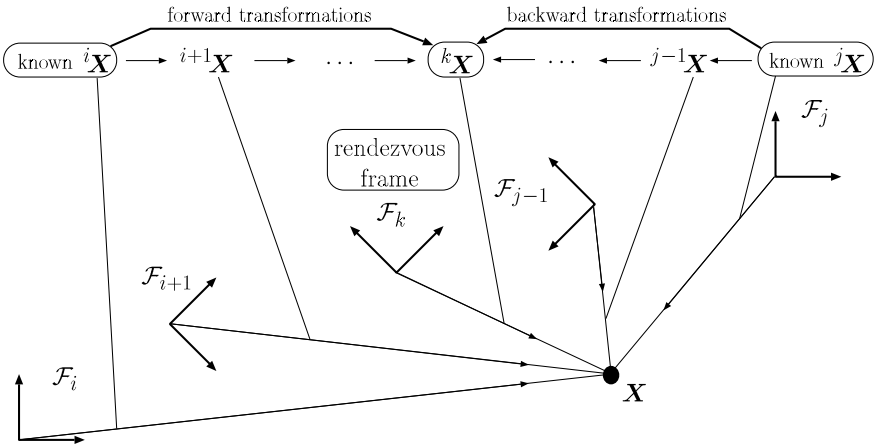


Fig. 10.8 Rendezvous method: If $^{i}\boldsymbol{X}$ and $^{j}\boldsymbol{X}$ are known, we can compute $^{k}\boldsymbol{X}$ for each $i \leq k \leq j$ in one of two different ways—(1) by successive forward transformations of $^{i}\boldsymbol{X}$, and (2) by successive backward transformation of $^{j}\boldsymbol{X}$

$$^{6}\boldsymbol{P}_3^o = {}^{6}\boldsymbol{M}_3 \; {}^{3}\boldsymbol{P}_3^o \; {}^{6}\widetilde{\boldsymbol{M}}_3 = 1 - I \begin{pmatrix} 0 \\ 0 \\ d_6 \end{pmatrix}. \tag{10.21}$$

Now we can compute $^{0}\boldsymbol{P}_3^o$ by

$$\begin{aligned} {}^{0}\boldsymbol{P}_3^o &= {}^{0}\boldsymbol{M}_6 \; {}^{6}\boldsymbol{P}_3^o \; {}^{0}\widetilde{\boldsymbol{M}}_6 = {}^{0}\boldsymbol{M}_6 \left(1 - I \begin{pmatrix} 0 \\ 0 \\ d_6 \end{pmatrix} \right) {}^{0}\widetilde{\boldsymbol{M}}_6 \\ &= 1 + I \begin{pmatrix} P_x \\ P_y \\ P_z \end{pmatrix}. \end{aligned} \tag{10.22}$$

Note that $^{0}\boldsymbol{M}_6$ is given. The vector $(P_x, P_y, P_z)^T$ describes the position of the origin $\boldsymbol{P}_3^o$ of frame $\mathcal{F}_3$ in frame $\mathcal{F}_0$ for a given overall transformation $^{0}\boldsymbol{M}_6$. Now we can apply the rendezvous method, since we know the representation of $\boldsymbol{P}_3^o$ in the two different frames $\mathcal{F}_0$ and $\mathcal{F}_3$ (see Fig. 10.9).

By applying successive forward transformations, we obtain

$$\begin{aligned} {}^{1}\boldsymbol{P}_3^o &= {}^{1}\boldsymbol{M}_0 \; {}^{0}\boldsymbol{P}_3^o \; {}^{1}\widetilde{\boldsymbol{M}}_0 \,, \\ {}^{2}\boldsymbol{P}_3^o &= {}^{2}\boldsymbol{M}_1 \; {}^{1}\boldsymbol{P}_3^o \; {}^{2}\widetilde{\boldsymbol{M}}_1 \,, \\ {}^{3}\boldsymbol{P}_3^o &= {}^{3}\boldsymbol{M}_2 \; {}^{2}\boldsymbol{P}_3^o \; {}^{3}\widetilde{\boldsymbol{M}}_2. \end{aligned} \tag{10.23}$$

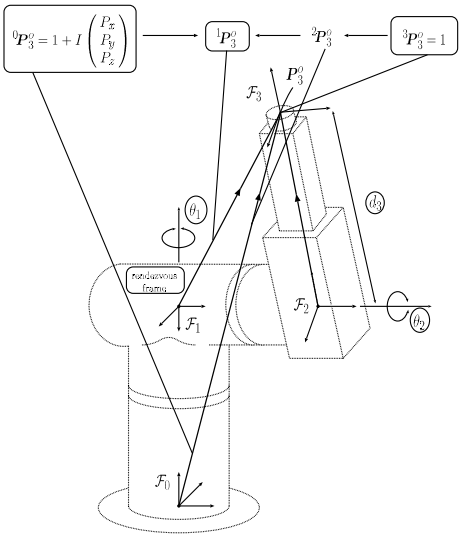


Fig. 10.9 The rendezvous method applied to $\boldsymbol{P}_3^o$ in order to determine the equations shown in Table 10.2 (the equations of rendezvous frame $\mathcal{F}_1$ are chosen to compute the variable parameters θ_1, θ_2, and d_3)

These computations were carried out with our Maple program, which calculated the left-hand sides of the four groups of equations in Table 10.2.

On the other hand, by applying successive backward transformations to the origin of $\mathcal{F}_3$ given by

$$^3\boldsymbol{P}_3^o = 1 + I \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} = 1, \tag{10.24}$$

we get

$$\begin{aligned} ^2\boldsymbol{P}_3^o &= {}^2\boldsymbol{M}_3 \; {}^3\boldsymbol{P}_3^o \; {}^2\widetilde{\boldsymbol{M}}_3 = 1 + I \begin{pmatrix} 0 \\ 0 \\ d_3 \end{pmatrix}, \\ ^1\boldsymbol{P}_3^o &= {}^1\boldsymbol{M}_2 \; {}^2\boldsymbol{P}_3^o \; {}^1\widetilde{\boldsymbol{M}}_2 = 1 + I \begin{pmatrix} d_3 \sin(\theta_2) \\ -d_3 \cos(\theta_2) \\ d_2 \end{pmatrix}, \\ ^0\boldsymbol{P}_3^o &= {}^0\boldsymbol{M}_1 \; {}^1\boldsymbol{P}_3^o \; {}^0\widetilde{\boldsymbol{M}}_1 = 1 + I \begin{pmatrix} d_3 \sin(\theta_2)\cos(\theta_1) - d_2 \sin(\theta_1) \\ d_3 \sin(\theta_2)\sin(\theta_1) + d_2 \cos(\theta_1) \\ d_3 \cos(\theta_2) + d_1 \end{pmatrix}. \end{aligned} \tag{10.25}$$

These equations correspond to the right-hand sides of the four groups of equations in Table 10.2. For simplicity, we have used the abbreviations s_i for $\sin(\theta_i)$ and c_i for $\cos(\theta_i)$.

Using the third equation of the rendezvous frame $\mathcal{F}_1$, we compute

Table 10.2 Rendezvous equations obtained for $\boldsymbol{P}_3^o$ regarding frames $\mathcal{F}_0$, $\mathcal{F}_1$, $\mathcal{F}_2$, and $\mathcal{F}_3$

Frame	Eq.	Forward		Backward
$\mathcal{F}_0$	1	P_x	=	$d_3 s_2 c_1 - d_2 s_1$
	2	P_y	=	$d_3 s_2 c_1 + d_2 c_1$
	3	P_z	=	$d_3 c_2 + d_1$
$\mathcal{F}_1$	1	$P_y s_1 + P_x c_1$	=	$d_3 s_2$
	2	$d_1 - P_z$	=	$-d_3 c_2$
	3	$P_y c_1 - P_x s_1$	=	d_2
$\mathcal{F}_2$	1	$-P_z s_2 + d_1 s_2 + P_x c_1 c_2 + P_y s_1 c_2$	=	0
	2	$d_2 - P_y c_1 + P_x s_1$	=	0
	3	$P_z c_2 - d_1 c_2 + P_x c_1 s_2 + P_y s_1 s_2$	=	d_3
$\mathcal{F}_3$	1	$-P_z s_2 + d_1 s_2 + P_x c_1 c_2 + P_y s_1 c_2$	=	0
	2	$d_2 - P_y c_1 + P_x s_1$	=	0
	3	$P_z c_2 - d_1 c_2 + P_x c_1 s_2 + P_y s_1 s_2 - d_3$	=	0

$$\theta_1 = \arctan_2(x_{1/2}, y_{1/2}), \tag{10.26}$$

where

$$x_{1/2} = \frac{d_2 - P_y y_{1/2}}{-P_x}, \qquad y_{1/2} = \frac{P_y d_2 \pm P_x \sqrt{P_x^2 + P_y^2 - d_2^2}}{P_x^2 + P_y^2}, \tag{10.27}$$

and

$$\arctan_2(x, y) = \begin{cases} \arctan\left(\frac{x}{y}\right) & : \quad y > 0 \\ \frac{\pi}{2} & : \quad y = 0 \text{ and } x > 0 \\ \text{undefined} & : \quad y = 0 \text{ and } x = 0 \\ -\frac{\pi}{2} & : \quad y = 0 \text{ and } x < 0 \\ \arctan\left(\frac{x}{y}\right) + \pi & : \quad y < 0\,. \end{cases} \tag{10.28}$$

This gives two values for θ_1. Now, let us look for d_3 and θ_2. For that, we consider the first and second equation of the rendezvous frame $\mathcal{F}_1$. With $a_{1/2} = P_y x_{1/2} + P_x y_{1/2}$ and $b = P_z - d_1$, we obtain two values for d_3. Since for the Stanford manipulator d_3 must be positive, we choose

$$d_{3_{1/2}} = \sqrt{a_{1/2}^2 + b^2}. \tag{10.29}$$

Using this value in equations 1 and 2, we compute straightforwardly,

$$\theta_2 = \arctan_2\left(\frac{a_{1/2}}{d_{3_{1/2}}}, \frac{b}{d_{3_{1/2}}}\right). \tag{10.30}$$

10.5.3 Computing θ_4 and θ_5 Using a Line

These variables will be computed using the joint transition from $\mathcal{F}_3$ to $\mathcal{F}_6$. Given the geometric characteristics of the manipulator, an appealing option is to use the line representation to set up an appropriate equation system. The representation ${}^0\boldsymbol{L}_6^z$ of the line $\boldsymbol{L}_6^z$ in frame $\mathcal{F}_0$ can be computed using ${}^0\boldsymbol{M}_6$:

$${}^0\boldsymbol{L}_6^z = {}^0\boldsymbol{M}_6\ {}^6\boldsymbol{L}_6^z\ {}^0\widetilde{\boldsymbol{M}}_6 = {}^0\boldsymbol{M}_6\ \left(\begin{pmatrix}0\\0\\1\end{pmatrix} + I\begin{pmatrix}0\\0\\0\end{pmatrix}\right)\ {}^0\widetilde{\boldsymbol{M}}_6. \tag{10.31}$$

Since the z-axis of the $\mathcal{F}_6$ frame crosses the origin of $\mathcal{F}_3$, we can see that the z-axis line related to this frame has zero moment. Thus, we can claim that L_6^z in the $\mathcal{F}_3$ frame is

$${}^3\boldsymbol{L}_6^z = {}^3\boldsymbol{M}_0\ {}^0\boldsymbol{L}_6^z\ {}^3\widetilde{\boldsymbol{M}}_0 = \begin{pmatrix}A_x\\A_y\\A_z\end{pmatrix} + I\begin{pmatrix}0\\0\\0\end{pmatrix}. \tag{10.32}$$

Note that ${}^3\boldsymbol{M}_0$ is known since we have already computed θ_1, θ_2, and d_3.

Now, applying successively forward transformations, as follows:

$$\begin{aligned}
{}^4\boldsymbol{L}_6^z &= {}^4\boldsymbol{M}_3\ {}^3\boldsymbol{L}_6^z\ {}^4\widetilde{\boldsymbol{M}}_3,\\
{}^5\boldsymbol{L}_6^z &= {}^5\boldsymbol{M}_4\ {}^4\boldsymbol{L}_6^z\ {}^5\widetilde{\boldsymbol{M}}_4,\\
{}^6\boldsymbol{L}_6^z &= {}^6\boldsymbol{M}_5\ {}^5\boldsymbol{L}_6^z\ {}^6\widetilde{\boldsymbol{M}}_5,
\end{aligned} \tag{10.33}$$

we obtain the left-hand sides of the four groups of equations in Table 10.3. The z-axis line $\boldsymbol{L}_6^z$ of $\mathcal{F}_6$ represented in $\mathcal{F}_6$ has zero moment and can therefore be expressed as

$${}^6L_6^z = \begin{pmatrix}0\\0\\1\end{pmatrix} + I\begin{pmatrix}0\\0\\0\end{pmatrix}. \tag{10.34}$$

Next, applying successive backward transformations, we obtain

$$\begin{aligned}
{}^5\boldsymbol{L}_6^z &= {}^5\boldsymbol{M}_6\ {}^6\boldsymbol{L}_6^z\ {}^5\widetilde{\boldsymbol{M}}_6,\\
{}^4\boldsymbol{L}_6^z &= {}^4\boldsymbol{M}_5\ {}^5\boldsymbol{L}_6^z\ {}^4\widetilde{\boldsymbol{M}}_5,\\
{}^3\boldsymbol{L}_6^z &= {}^3\boldsymbol{M}_4\ {}^4\boldsymbol{L}_6^z\ {}^3\widetilde{\boldsymbol{M}}_4.
\end{aligned} \tag{10.35}$$

Table 10.3 Rendezvous equations obtained for $\boldsymbol{L}_6^z$ regarding frames $\mathcal{F}_3$, $\mathcal{F}_4$, $\mathcal{F}_5$, and $\mathcal{F}_6$

Fr.	Eq.	Forward		Backward
$\mathcal{F}_3$	1	A_x	=	$-c_4 s_5$
	2	A_y	=	$-s_4 s_5$
	3	A_z	=	$-c_5$
$\mathcal{F}_4$	1	$A_y s_4 + A_x c_4$	=	$-s_5$
	2	A_z	=	$-c_5$
	3	$A_y c_4 - A_x s_4$	=	0
$\mathcal{F}_5$	1	$-A_z s_5 + A_x c_4 c_5 + A_y s_4 c_5 c_6$	=	0
	2	$A_y c_4 - A_x s_4$	=	0
	3	$-A_z c_5 - A_x c_4 s_5 - A_y s_4 s_5$	=	1
$\mathcal{F}_6$	1	$A_x s_4 s_6 - A_y c_4 s_6 + A_y s_4 c_5 c_6 + A_x c_4 c_5 c_6 - A_z s_5 c_6$	=	0
	2	$-A_x s_4 c_6 + A_y c_4 c_6 + A_y s_4 c_5 c_6 + A_x c_4 c_5 s_6 - A_z s_5 s_6$	=	0
	3	$-A_z c_5 - A_x c_4 s_5 - A_y s_4 s_5$	=	1

Using our Maple program, we then compute the right-hand sides of the four groups of equations in Table 10.3.

Finally, we will consider the equations of rendezvous frame $\mathcal{F}_4$. Using the third equation of the Table 10.3, we compute

$$\theta_4 = \arctan_2(x_{1/2}, y_{1/2}), \tag{10.36}$$

where

$$x_{1/2} = -\frac{A_y y_{1/2}}{-Ax} = \pm\frac{A_y}{\sqrt{A_x^2 + A_y^2}}, \qquad y_{1/2} = \pm\frac{A_x}{\sqrt{A_x^2 + A_y^2}}. \tag{10.37}$$

This results in two values for θ_4, which when substituted into the first and second equation of the Table 10.3 helps us to find two solutions for θ_5:

$$\theta_5 = \arctan_2(s_5, c_5) = \arctan_2\Big((-A_y s_4 - A_x c_4), -A_z\Big). \tag{10.38}$$

10.5.4 Computing θ_6 Using a Plane Representation

Since θ_1, θ_2, d_3, θ_4, and θ_5 are now known, we can compute the motor ${}^5\boldsymbol{M}_0$. The yz-plane $\boldsymbol{H}_6^{yz}$ represented in $\mathcal{F}_6$ has a Hesse distance of zero; thus,

$$ {}^6\boldsymbol{H}_6^{yz} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} + I0 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}. \tag{10.39} $$

The transformation of this plane to $\mathcal{F}_0$ reads

$$ {}^0\boldsymbol{H}_6^{yz} = {}^0\boldsymbol{M}_6 \; {}^6\boldsymbol{H}_6^{yz} \; {}^0\widetilde{\boldsymbol{M}}_6 = {}^0\boldsymbol{M}_6 \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} {}^0\widetilde{\boldsymbol{M}}_6. \tag{10.40} $$

Now we compute ${}^5\boldsymbol{H}_6^{yz}$:

$$ {}^5\boldsymbol{H}_6^{yz} = {}^5\boldsymbol{M}_0 \; {}^0\boldsymbol{H}_6^{yz} \; {}^5\widetilde{\boldsymbol{M}}_0 = \begin{pmatrix} N_x \\ N_y \\ N_z \end{pmatrix} + I \, {}^5d_{H_6^{yz}}. \tag{10.41} $$

The orientation bivector $(N_x, N_y, N_z)^T$ describes the orientation of the yz-plane of frame $\mathcal{F}_6$ in frame $\mathcal{F}_5$, given the values of the joint variables θ_1, θ_2, θ_4, θ_5, and d_3. By applying forward transformation from $\mathcal{F}_5$ to $\mathcal{F}_6$, we obtain

$$ {}^6\boldsymbol{H}_6^{yz} = {}^6\boldsymbol{M}_5 \; {}^5\boldsymbol{H}_6^{yz} \; {}^6\widetilde{\boldsymbol{M}}_5. \tag{10.42} $$

Then, using our Maple program, we calculate the left-hand sides of the two groups of equations in Table 10.4. Since the values for θ_1, θ_2, d_3, θ_4, and θ_5 are not unique, we will obtain different values for the equations. By applying ${}^5\boldsymbol{M}_6$ to ${}^6\boldsymbol{H}_6^{yz}$, we next obtain the right-hand sides of the two groups of equations in Table 10.4:

Table 10.4 Rendezvous equations obtained for $\boldsymbol{H}_6^{yz}$ regarding frames $\mathcal{F}_5$ and $\mathcal{F}_6$

Frame	Eq.	Forward		Backward
$\mathcal{F}_5$	1	N_x	=	c_6
	2	N_y	=	s_6
	3	N_z	=	0
$\mathcal{F}_6$	1	$N_y s_6 + N_x c_6$	=	1
	2	$N_x s_6 - N_y c_6$	=	0
	3	N_z	=	0

$$
\begin{aligned}
{}^5\boldsymbol{H}_6^{yz} &= {}^5\boldsymbol{M}_6\ {}^6\boldsymbol{H}_6^{yz}\ {}^5\widetilde{\boldsymbol{M}}_6 = {}^5\boldsymbol{M}_6 \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} {}^5\widetilde{\boldsymbol{M}}_6, \\
&= \begin{pmatrix} \sin(\theta_6) \\ \cos(\theta_6) \\ 0 \end{pmatrix}. \qquad (10.43)
\end{aligned}
$$

Finally, we will consider the equations of the rendezvous frame $\mathcal{F}_5$. Using the first and second equations in the Table 10.4, we can compute θ_6 by

$$\theta_6 = \arctan_2(s_6, c_6) = \arctan_2(N_x, N_y). \qquad (10.44)$$

Note that since there are two values for θ_4 and two values for θ_5, there is more than one solution for θ_6.

10.6 Inverse Kinematic Using the 3D Affine Plane

In this subsection, we compute the inverse kinematic of a robot manipulator using the framework of the 3D affine plane. On the one hand, one can use a geometric algebra $\mathcal{G}_{p,q,0}$ with Minkowski metric for computations of projective geometry and the algebra of incidence, as is the case for computer vision problems. On the other hand, one can use a degenerated algebra for computations involving rigid motions, as is often the case in the field of robotics (see, e.g., our use of the motor algebra $\mathcal{G}^+_{3,0,1}$ in Sect. 10.5 or the Clifford algebra $\mathcal{G}_{3,0,1}$ in [3]). In a more general way, the 3D affine plane allows calculations which can involve both 3D rigid transformations and the meet and join operations of the algebra of incidence.

Note that in this section, we assume that the projection P_A back to the affine plane is always carried out, and thus we need not make it explicit in the formulas. In the procedure, after the equations to compute the angles θ_i have been found, simple dual trigonometric relations can then be used to determine the angles. In order to show the methodology at a symbolic level, we will avoid showing these trigonometric computation details.

The transformation M_t of a robot manipulator which takes the end-effector from its home position to a configuration determined by the n-degrees of freedom of the joint angles $\theta_1, \theta_2, \ldots, \theta_n$ is given by

$$M_t = M_1 M_2 M_3 \ldots M_n, \qquad (10.45)$$

where the screw versor of a joint $M_i = T_i R_i$ is dependent on the angle θ_i.

The task of *inverse kinematics* is to calculate the angles θ_i for a given final configuration of the end-effector. Robot manipulators are equipped with a single parallel revoluted axis as well as intersecting axes. The latter can be located at the

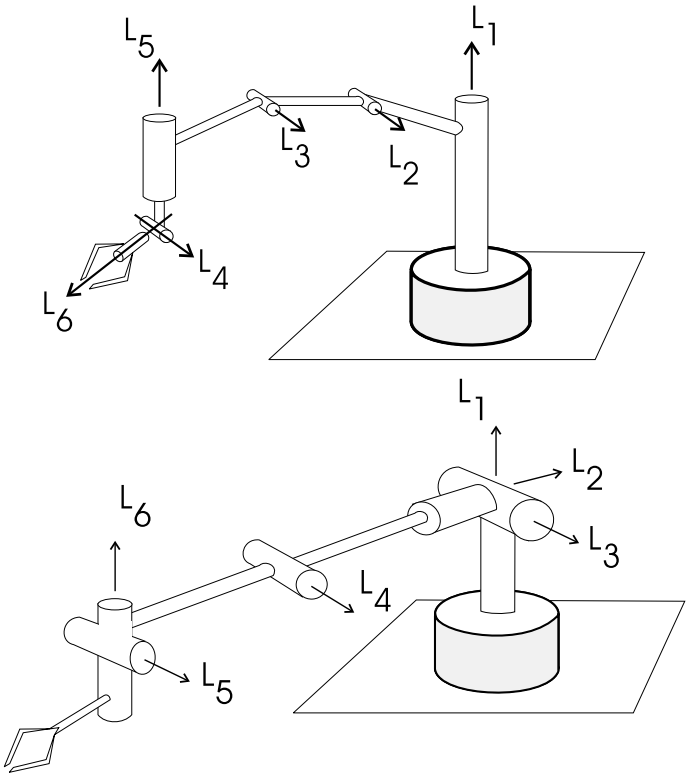


Fig. 10.10 Robot manipulators: **a** (top) intersect revoluted line axes at the end-effector; or **b** (bottom) at the home position

end-effector or at the home position. Two typical configurations are illustrated in Fig. 10.10a–b. The mechanical characteristics of the robot manipulators can be used to simplify the computations—by considering either the invariant plane ϕ^h, in the case of three parallel revoluted line axes (see Fig. 10.10a), or the invariant point p^h, in the case of the intersecting revoluted line axis (see Fig. 10.10b).

We can solve the inverse kinematics problem by breaking it up into a series of separate equations, using the strategy suggested by Selig [3] (see Chap. 10). We will illustrate the procedure for a robot with 6 degrees of freedom. First, we rearrange the terms of Eq. (10.45):

$$M_2 M_3 M_4 = \widetilde{M}_1 M_t \widetilde{M}_6 \widetilde{M}_5. \tag{10.46}$$

In the case of three parallel joints, we can isolate them by considering the common perpendicular plane ϕ^h which satisfies the equation

$$\phi^h = M_2 M_3 M_4 \, \phi^h \, \widetilde{M}_4 \widetilde{M}_3 \widetilde{M}_2 = \widetilde{M}_1 M_t (\widetilde{M}_6 (\widetilde{M}_5 \, \phi^h \, M_5) M_6) \widetilde{M}_t M_1. \tag{10.47}$$

In the case of a meeting point p^h, we can isolate the three coincident joints as follows:

$$p^h = M_2M_3M_4\,p^h\,\widetilde{M}_4\widetilde{M}_3\widetilde{M}_2 = \widetilde{M}_1M_t\widetilde{M}_6\widetilde{M}_5\,p^h\,M_5\widetilde{M}_6\widetilde{M}_tM_1. \tag{10.48}$$

In this way, we have separated the problem into two systems of equations:

$$\widetilde{M}_tM_1\,\phi^h\,\widetilde{M}_1M_t = \widetilde{M}_6\widetilde{M}_5\,\phi^h\,M_5M_6 \tag{10.49}$$

or

$$\widetilde{M}_tM_1\,p^h\,\widetilde{M}_1M_t = \widetilde{M}_6\widetilde{M}_5\,p^h\,M_5M_6, \tag{10.50}$$

$$M_2M_3M_4 = M_1M_t\widetilde{M}_6\widetilde{M}_5 = M'. \tag{10.51}$$

We can now compute for $\theta_1,\ \theta_5,\ \theta_6$ with the help of either Eq. (10.49) (see Fig. 10.10a) or Eq. (10.50) (see Fig. 10.10b). Then, by using these results and Eq. (10.51), we can solve for $\theta_2,\ \theta_3,\ \theta_4$.

Let us demonstrate how the procedure works for the case of three intersecting revoluted joint axes in the common plane at the end-effector (see Fig. 10.10a). When the plane ϕ_b^h (perpendicular to the line axes l_2, l_3, and l_4) is rotated about the end-joint, the point p_i^h on the line axis of the revoluted end-joint l_6^h remains invariant. Using a *meet* operation and Eq. (10.49), the angle θ_6 can be eliminated:

$$\begin{aligned} p_i^h &= (\widetilde{M}_tM_1\,\phi^h\,\widetilde{M}_1M_t) \cap l_6^h = (\widetilde{M}_6\widetilde{M}_5\,\phi^h\,M_5M_6) \cap l_6^h, \\ &= (\widetilde{M}_5\,\phi^h\,M_5) \cap l_6^h. \end{aligned} \tag{10.52}$$

In the case of a robot manipulator in the home position, the revoluted joint axes are the manipulator basis. Equation (10.50) shows that the point p^h is an invariant for the parallel fourth and fifth line axes. Thus, we can use the equation

$$M_6\widetilde{M}_t\,p^h\,M_t\widetilde{M}_6 = \widetilde{M}_5\widetilde{M}_4\,p^h\,M_4M_5 \tag{10.53}$$

to solve for the angles θ_4 and θ_5. Using the line l_5 and p^h, we get the invariant plane,

$$\phi_i^h = M_6\widetilde{M}_t\,p^h\,M_t\widetilde{M}_6 \wedge l_5. \tag{10.54}$$

The 3D coordinates of this plane are known in advance and correspond to the plane $x - z$ or e_{32}; thus, Eq. (10.54) allows us to solve for the angle θ_6. Having determined angle θ_6, we can now use Eq. (10.53) to complete the calculation of θ_4 and θ_5.

Now consider the three coincident line axes $l_1^h,\ l_2^h,\ l_3^h$ given in Fig. 10.10b. We can isolate the angle θ_2 by considering the invariant relation based on the meet of two of these lines. Thus,

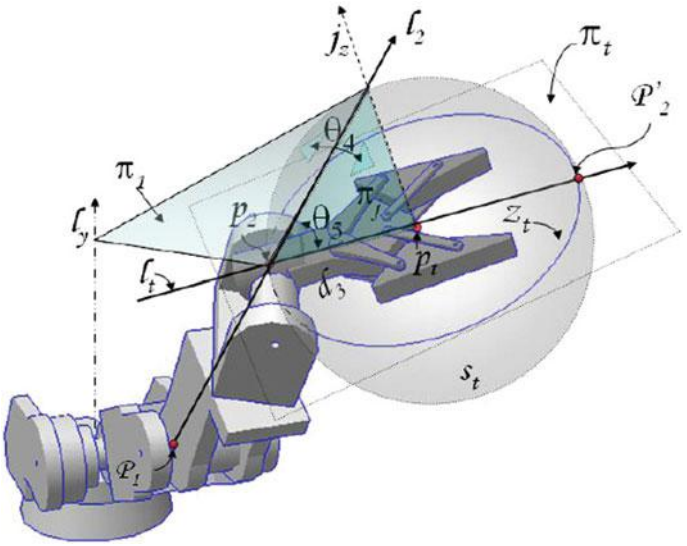
$$(M_2\,l_3^h\,\widetilde{M}_2) \cap l_1^h = (M'\,l_3^h\,\widetilde{M}') \cap l_1^h = p_0^h, \tag{10.55}$$

where $M' = M_1M_2M_3$ and p_0^h is the invariant intersecting point. When the lines are parallel, as shown in Fig. 10.10a, we can use the same invariant relation, but now we consider the intersecting point to be at infinity, so that $M' = M_2M_3M_4$.

10.7 Inverse Kinematic Using Conformal Geometric Algebra

In this subsection, we will briefly describe the procedure steps to compute the inverse kinematics of a 5-DOF robot arm using the conformal geometric algebra framework. We have presented above similar computation using motor algebra $\mathcal{G}_{3,0,1}$ and the affine plane and next conformal geometric algebra $\mathcal{G}_{4,1}$. Our purpose is that the reader by studying these examples learns to compute using the geometric algebra and he or she can get a better understanding of the potential and limitations of the different computational framework models of geometric algebra.

Objective: Find the joint angles in order to place the robot arm at the point $\boldsymbol{p}_t$ in such a way that the gripper will stay parallel to the plane ϕ_t.



Step 1: Find the position of the point $\boldsymbol{p}_2$

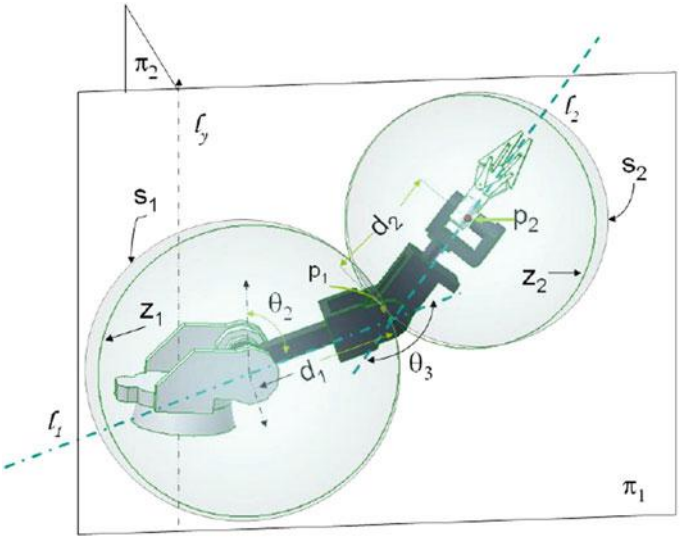
$$
\begin{aligned}
\boldsymbol{l}_y^* &= e_2\boldsymbol{E}, \\
\boldsymbol{s}_t &= \boldsymbol{p}_t - \frac{1}{2}d_3^2 e_\infty, \\
\boldsymbol{z}_t &= \boldsymbol{s}_t \wedge \pi_t, \\
\boldsymbol{j}_z^* &= \boldsymbol{z}_t \wedge e_\infty, \\
\boldsymbol{l}_d^* &= d(\boldsymbol{p}_t, \boldsymbol{l}_y^*) \wedge \boldsymbol{p}_t \wedge e_\infty,
\end{aligned}
$$

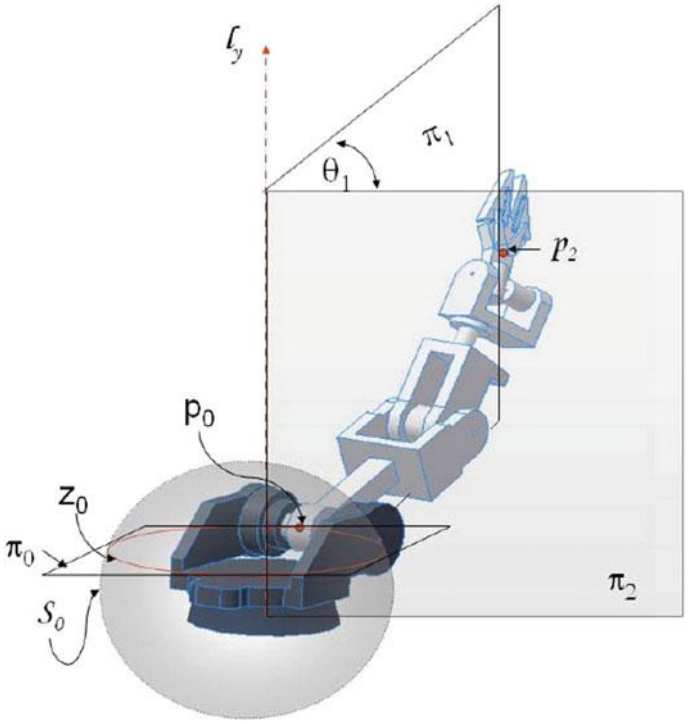
$$\begin{aligned}
\pi^*_{jl_d} &= \boldsymbol{j}^*_z \wedge (\boldsymbol{l}^*_d \boldsymbol{E}),\\
\boldsymbol{P}_{P_2} &= \pi_{jl_d} \wedge \boldsymbol{z}_t,\\
\boldsymbol{p}_2 &= \frac{\boldsymbol{P}^*_{P_2} + \sqrt{\boldsymbol{P}^*_{P_2} \cdot \boldsymbol{P}^*_{P_2}}}{\boldsymbol{P}^*_{P_2} \cdot e_\infty},
\end{aligned}$$

where $d(\boldsymbol{p}_t, \boldsymbol{l}^*_y)$ stands for the directed distance between $\boldsymbol{p}_t$ and $\boldsymbol{l}^*_y$.

Step 2: Find the point $\boldsymbol{p}_1$

$$\begin{aligned}
\boldsymbol{s}_1 &= e_0 - \frac{d_1^2}{2} e_\infty,\\
\boldsymbol{s}_2 &= \boldsymbol{p}_2 - \frac{d_2^2}{2} e_\infty,\\
\pi^*_1 &= \boldsymbol{l}_y \wedge \boldsymbol{p}_2 = e_2 \boldsymbol{E} \wedge \boldsymbol{p}_2,\\
\pi^*_2 &= e_3 I_c,\\
\boldsymbol{P}_{P_1} &= \boldsymbol{s}_1 \wedge \boldsymbol{s}_2 \wedge \pi_1,\\
\boldsymbol{p}_1 &= \frac{\boldsymbol{P}^*_{P_1} + \sqrt{\boldsymbol{P}^*_{P_1} \cdot \boldsymbol{P}^*_{P_1}}}{\boldsymbol{P}^*_{P_1} \cdot e_\infty}.
\end{aligned}$$





Step 3: Find the point $\boldsymbol{p}_0$

$$\begin{aligned}
\pi_1^* &= \boldsymbol{l}_y^* \wedge \boldsymbol{p}_2, \\
\boldsymbol{s}_0 &= e_0 - \frac{d_0^2}{2} e_\infty, \\
\pi_0 &= e_2 - h e_\infty, \\
\boldsymbol{z}_0 &= \boldsymbol{s}_0 \wedge \pi_0, \\
\boldsymbol{P}_{P_0} &= \boldsymbol{z}_0 \wedge \pi_1, \\
\boldsymbol{p}_0 &= \frac{\boldsymbol{P}_{P_0}^* + \sqrt{\boldsymbol{P}_{P_0}^* \cdot \boldsymbol{P}_{P_0}^*}}{\boldsymbol{P}_{P_0}^* \cdot e_\infty}.
\end{aligned}$$

Step 4: Compute the lines $\boldsymbol{l}_1$, $\boldsymbol{l}_2$, and $\boldsymbol{l}_3$

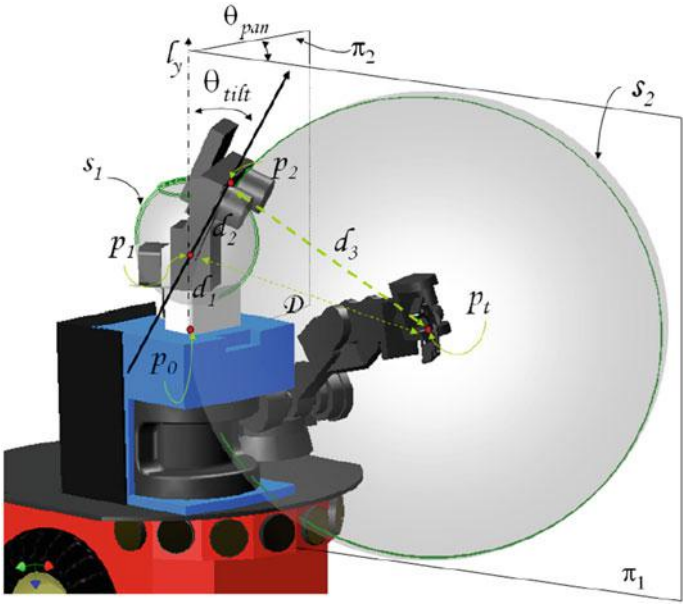
$$\begin{aligned}
\boldsymbol{l}_1^* &= \boldsymbol{p}_0 \wedge \boldsymbol{p}_1 \wedge e_\infty, \\
\boldsymbol{l}_2^* &= \boldsymbol{p}_1 \wedge \boldsymbol{p}_2 \wedge e_\infty, \\
\boldsymbol{l}_3^* &= \boldsymbol{p}_2 \wedge \boldsymbol{p}_1 \wedge e_\infty.
\end{aligned} \tag{10.56}$$

Step 5: Compute the angles

$$\cos(\theta_1) = \frac{\pi_1^* \cdot \pi_2^*}{|\pi_1^*||\pi_2^*|}, \quad \cos(\theta_2) = \frac{\boldsymbol{l}_1^* \cdot \boldsymbol{l}_y^*}{|\boldsymbol{l}_1^*||\boldsymbol{l}_y^*|},$$
$$\cos(\theta_3) = \frac{\boldsymbol{l}_1^* \cdot \boldsymbol{l}_3^*}{|\boldsymbol{l}_1^*||\boldsymbol{l}_3^*|}, \quad \cos(\theta_4) = \frac{\pi_1^* \cdot \pi_3^*}{|\pi_1^*||\pi_3^*|},$$
$$\cos(\theta_5) = \frac{\boldsymbol{l}_3^* \cdot \boldsymbol{l}_2^*}{|\boldsymbol{l}_3^*||\boldsymbol{l}_2^*|}.$$

Next, we present a procedure involving the computing of the inverse kinematics of a Pan–Tilt Unit (PTU).

Objective: Find the joint angles of the PTU unit in order to orient its principal ray toward the target point $\boldsymbol{p}_t$.



Step 1: Find $\boldsymbol{p}_2$

$$\begin{aligned}
\boldsymbol{s}_1 &= \boldsymbol{p}_1 - \frac{1}{2} d_2^2 e_\infty, \\
D &= d(\boldsymbol{p}_1, \boldsymbol{p}_t), \\
\boldsymbol{s}_2 &= \boldsymbol{p}_1 - \frac{1}{2}(D - d_2^2) e_\infty, \\
\boldsymbol{l}_y^* &= \boldsymbol{p}_0 \wedge \boldsymbol{p}_1 \wedge e_\infty, \\
\pi_1^* &= \boldsymbol{l}_y^* \wedge \boldsymbol{p}_t, \\
\boldsymbol{PP}_2 &= \boldsymbol{s}_1 \wedge \boldsymbol{s}_2 \wedge \pi_1.
\end{aligned}$$

Step 2: Compute line and plane

$$\boldsymbol{l}_2^* = \boldsymbol{p}_1 \wedge \boldsymbol{p}_2 \wedge e_\infty,$$
$$\pi_2^* = \boldsymbol{l}_y^* \wedge e_3.$$

Step 3: Compute tilt and pan angles

$$\cos(\theta_{tilt}) = \frac{\boldsymbol{l}_y \cdot \boldsymbol{l}_2}{|\boldsymbol{l}_y||\boldsymbol{l}_2|},$$
$$\cos(\theta_{pan}) = \frac{\pi_1 \cdot \pi_2}{|\pi_1||\pi_2|},$$

where $d(\boldsymbol{p}_1, \boldsymbol{p}_t)$ stands for the directed distance between $\boldsymbol{p}_1$ and $\boldsymbol{p}_t$.

10.8 Conclusion

This chapter presented the application of the algebra of motors for the treatment of the direct and inverse kinematics of robot manipulators. When dealing with 3D rigid motion, it is usual to use homogeneous coordinates in the 4D space to linearize this nonlinear 3D transformation. With the same effect, we model the prismatic and revolute motion of points, lines, and planes using motors which are equivalent to screws. In our approach, we can also use the representation of planes, and this expands the useful geometric language for the treatment of robotic problems. In addition, we illustrate the use of the 3D affine plane for computing inverse kinematics, taking advantage of incidence algebra operations to simplify computations that use intersections and unions of points, lines, and planes under rigid motion.

The chapter has shown the flexibility of the motor algebra approach for the solution of the direct and inverse kinematics of robot manipulators. To solve the robot's inverse kinematics, we have shown that, depending on our needs, we can use representations either of points, lines, or planes, which help to enormously reduce the complexity of the computation without losing geometric insight. Similarly, the use of incidence algebra in the affine plane framework allows us to define geometric constraints, or flags, by taking advantage of meet and joint operations, thereby allowing easy identification of the most important geometric objects involved.

For the computation of inverse kinematics, we have used also the conformal geometric algebra which besides points, lines, and planes uses circles and spheres as well, and the use of this broad set of geometric entities helps us greatly to simplify the complex computation of the inverse kinematics.

The main contribution of this chapter has been to show that our approach offers more flexibility while at the same time preserving geometric insight during computation. The author believes that the versatility of the geometric algebra framework offers an alternative approach to the increasingly complex of application of multi-link mechanisms.

References

1. Denavit, J., & Hartenberg, R. S. (1955). A kinematic notation for the lower-pair mechanism based on matrices. *Journal of the Optical Society of America, A*, 465–484.
2. Bayro-Corrochano, E., Daniilidis, K., & Sommer, G. (1997). Hand-eye calibration in terms of motions of lines using geometric algebra. In *10th Scandinavian Conference on Image Analysis* (Vol. I, pp. 397–404). Finland: Lappeenranta.
3. Selig, J. M. (1999). Clifford algebra of points, lines and planes. Technical report SBU-CISM-99-06, South Bank University, School of Computing, Information Technology and Maths.

Chapter 11
Robot Dynamics

11.1 Introduction

The study of the kinematics and dynamics of robot mechanisms has employed different frameworks, such as vector calculus, quaternion algebra, or linear algebra; the last is used most often. However, in these frameworks, handling the kinematics and dynamics involving only points and lines is very complicated. In previous chapter, the motor algebra was used to treat the kinematics of robot manipulators using the points, lines, and planes. We also used the conformal geometric algebra which includes for the representation also circles and spheres. The use of additional geometric entities helps even more to reduce the representation and computational difficulties. In this chapter, we will show that the mathematical treatment of dynamics indeed becomes much easier using the geometric primitives and spinors of the conformal geometric algebra framework. Selig [[1], Chap. 9] studies the Newton mechanics and the Newton–Euler recursive algorithm using screws as vectors and the transformations using matrices. This chapter studies the Newton mechanics and the Newton–Euler recursive algorithm using the geometric algebra framework. The Euler–Lagrange dynamics and the Hamiltonian mechanics on phase space are formulated in the geometric algebra as well.

In this chapter, we use bold for vectors and slant bold for bivectors. Since in motor algebra $G^{+}_{3,0,1}$ or conformal geometric algebra $G_{4,1}$, we can't model the inertia matrix $\mathtt{N}$ as spinor, we use for the inertia transformation of screws the tensor $\mathtt{N}(\cdot)$.

11.2 D'Alembert's Principle

D'Alembert's principle is also known the Lagrange–d'Alembert principle, and is a statement of the fundamental classical laws of motion. The French physicist and mathematician Jean le Rond d'Alembert named so this principle after its discovery. It can be shown that it is the dynamic analog to the principle of virtual work for

© Springer Nature Switzerland AG 2020

E. Bayro-Corrochano, *Geometric Algebra Applications Vol. II*,
https://doi.org/10.1007/978-3-030-34978-3_11

applied forces in a static system. As a matter of fact is indeed more general than Hamilton's principle, avoiding restriction to holonomic systems [[2], p. 92]. Note that a holonomic constraint depends only on the coordinates and time, it does not depend on the velocities. If the negative terms in accelerations are seen as inertial forces, the statement of d'Alembert's principle turns to be *the total virtual work* of the exerted forces plus the inertial forces vanish for reversible displacements [[2], p. 90]. This principle does not apply for irreversible displacements, such as sliding friction, where a more general specification of the irreversibility would be needed [3].

The principle states that the sum of the differences between the forces acting on a system of mass particles and the time derivatives of the momenta of the system itself projected onto any virtual displacement consistent with the constraints of the system is zero. Thus, one can formulate d'Alembert's principle which states that the difference of applied forces and inertial forces for a dynamic system does no virtual work

$$\delta W = \sum_{i=1}^{N} (\mathbf{F}_i - m_i \mathbf{a}_i) \cdot \delta \mathbf{r}_i = 0, \tag{11.1}$$

where i indices a particular particle in the system, $\mathbf{F}_i$ is the total applied force (excluding constraint forces) on the i-th particle, m_i is the mass of the i-th particle, $\mathbf{a}_i$ is the acceleration of the i-th particle, $m_i \mathbf{a}_i$ together as product represents the time derivative of the momentum of the i-th particle if the masses do not change with time, and $\delta \mathbf{r}_i$ is the virtual displacement of the i-th particle, consistent with the constraints.

Equation (11.1) is often called d'Alembert's principle, but surprisingly it was first written in this variational form by Joseph Louis Lagrange. So, d'Alembert's contribution was simply to demonstrate that in the totality of a dynamic system the forces of constraint vanish. Better to say, that the generalized forces $\mathbf{Q}_j$ need not include constraint forces. Furthermore, we can see it as equivalent to the somewhat more cumbersome Gauss's principle of least constraint.

11.3 Newtonian Mechanics

In this section, we will study the Newtonian mechanics of manipulator arms. We will consider the simplest possible model for robots which consists of several rigid bodies, the links, connected together. We will ignore such effects as friction in the joints, flexibility of the joints and links, and also the dynamics of the motors and drives themselves.

From Newton's laws of mechanics, the motion equation of a single particle in terms of bivectors reads

$$\boldsymbol{f} = m\dot{\boldsymbol{v}}, \tag{11.2}$$

where $\boldsymbol{f}$ is the force exerted on the particle, m is its mass, and $\dot{\boldsymbol{v}} = \frac{\mathrm{d}^2\boldsymbol{x}}{\mathrm{d}t^2}$. Another more general equation for the force involving the rate of change of the momentum $\boldsymbol{p}$ reads

$$\boldsymbol{f} = \frac{\mathrm{d}\boldsymbol{p}}{\mathrm{d}t}. \tag{11.3}$$

Note that these two equations are the same when the mass do not change, since the momentum is given by

$$\boldsymbol{p} = m\boldsymbol{v}. \tag{11.4}$$

This equation tells linear momentum is mass times velocity. On the other hand, the angular momentum τ of the particle is defined by

$$\boldsymbol{\tau} = \boldsymbol{x} \wedge \boldsymbol{p}, \tag{11.5}$$

and thus its rate of change is given by

$$\frac{\mathrm{d}\boldsymbol{\tau}}{\mathrm{d}t} = \frac{\mathrm{d}\boldsymbol{x} \wedge \boldsymbol{p}}{\mathrm{d}t} = \boldsymbol{x} \wedge \boldsymbol{f}, \tag{11.6}$$

where

$$\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t} \wedge \boldsymbol{p} = m\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t} \wedge \frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t} = 0. \tag{11.7}$$

Equation (11.6) tells us that the moment of the applied force equals the rate of change of angular momentum.

A rigid body can be seen as a collection of particles all held together by internal forces. The effects of these internal forces always cancel, thus they can be ignored. The total mass m of the body is the sum of the masses of all particles. Since the particles are so small, the total mass can be calculated using the volume integral $\mathrm{d}v$ and the density of the body ρ, i.e.,

$$m = \int \rho \mathrm{d}v. \tag{11.8}$$

The center of mass of the body is a point where the weight appears to act and it can be computed as follows:

$$\boldsymbol{c} = \frac{1}{m}\int \rho\boldsymbol{x}\mathrm{d}v. \tag{11.9}$$

If we assume that the density and the mass of the body are uniform, the total linear momentum is computed as follows:

$$\boldsymbol{p}_{tot} = \int \rho \frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t} \mathrm{d}v = \frac{\mathrm{d}}{\mathrm{d}t} \int \rho \boldsymbol{x} \mathrm{d}v = m \frac{\mathrm{d}\boldsymbol{c}}{\mathrm{d}t}. \tag{11.10}$$

If we consider the forces on the body, the body can be replaced by a single particle of mass m located at the center of the mass with a total force as

$$\boldsymbol{f}_{tot} = m \frac{\mathrm{d}^2 \boldsymbol{c}}{\mathrm{d}t^2}, \tag{11.11}$$

which corresponds to differentiate the total momentum $\boldsymbol{p}_{tot}$. Note that slant bold is used to denote a bivector and italics for a 3×3 column vector, e.g., $\boldsymbol{\omega} = \omega_x e_{23} + \omega_y e_{31} + \omega_z e_{12}$ or $\omega = [\omega_x, \omega_y, \omega_z]^T$.

Given a twist $\boldsymbol{s} = \boldsymbol{\omega} + e_\infty \boldsymbol{v}$ in $G_{4,1}$, the velocity at any point in the body is $\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t} = \boldsymbol{\omega} \wedge \boldsymbol{x} + \boldsymbol{v}$, if we apply this equation to the center of mass, the total linear momentum is given by

$$\boldsymbol{p}_{tot} = m\boldsymbol{\omega} \wedge \boldsymbol{c} + m\boldsymbol{v}. \tag{11.12}$$

Similarly, we can express the total angular momentum as the volume integral

$$\boldsymbol{j}_{total} = \int \rho \boldsymbol{x} \wedge \frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t} \mathrm{d}v. \tag{11.13}$$

If we consider that the body is performing an instantaneous twist, we can replace $\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t}$ with $\boldsymbol{w} \wedge \boldsymbol{x} + \boldsymbol{v}$, thus the total angular momentum reads

$$\boldsymbol{j}_{tot} = \sum \{\rho \boldsymbol{x} \wedge (\boldsymbol{\omega} \wedge \boldsymbol{x}) + \rho \boldsymbol{x} \wedge \boldsymbol{v}\} \mathrm{d}v = \int \rho \boldsymbol{x} \wedge (\boldsymbol{\omega} \wedge \boldsymbol{x}) \mathrm{d}v + m\boldsymbol{c} \wedge \boldsymbol{v}. \tag{11.14}$$

Note that the integral term contains the triple wedge product $\boldsymbol{x} \wedge (\boldsymbol{\omega} \wedge \boldsymbol{x})$ and if $\boldsymbol{x} = xe_{23} + ye_{31} + ze_{12}$ and $\boldsymbol{\omega} = \omega_x e_{23} + \omega_y e_{31} + \omega_z e_{12}$

$$\begin{aligned}
&\boldsymbol{x} \wedge (\boldsymbol{\omega} \wedge \boldsymbol{x}) = ((y^2 + z^2)\omega_x - xy\omega_y - xz\omega_z)e_{23} + \\
&(-xy\omega_x + (x^2 + z^2)\omega_y - yz\omega_z)e_{31} + (-xz\omega_x - yz\omega_y + (x^2 + y^2)\omega_z)e_{12} \\
&\equiv \begin{bmatrix} (y^2 + z^2) & -xy & -xz \\ -xy & (x^2 + z^2) & -yz \\ -xz & -yz & (x^2 + y^2) \end{bmatrix} \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}.
\end{aligned} \tag{11.15}$$

Thus, we can write the total angular momentum as follows:

$$\boldsymbol{j}_{total} = \mathtt{I}(\boldsymbol{\omega}) + m\boldsymbol{c} \wedge \boldsymbol{v}, \tag{11.16}$$

where $\mathtt{I}$ is symmetric and it is equivalent to the action of the 3×3 inertia matrix of the body. Its entries depend only on the mass distribution of the body and they are given by

$$\begin{bmatrix} \int \rho(y^2+z^2)dv & -\int \rho xy dv & -\int \rho xz dv \\ -\int \rho xy dv & \int \rho(x^2+z^2)dv & -\int \rho yz dv \\ -\int \rho xz dv & -\int \rho yz dv & \int \rho(x^2+y^2)dv \end{bmatrix}. \tag{11.17}$$

Be aware that the inertia matrix $\mathtt{I}$ changes as the body moves. The matrix $\mathtt{I}$ applied to the bivectors is a function and for simplicity we denote as $\mathtt{I}(\cdot)$ acting on the bivector. Its actual computing will be done using a vector representation of the bivector, i.e.,

$$\mathtt{I}(\boldsymbol{w}) \equiv \mathtt{I} \begin{bmatrix} w_x \\ w_y \\ w_z \end{bmatrix}. \tag{11.18}$$

Since $G_{4,1}$ or $G^+_{3,0,1}$ are not equipped with bivectors to represent these matrices as spinors, we use simply as a function to denote these transformations. Equation (7.107) of Sect. 7.4 shows how in $G_{9,3}$ one can represent the inertia matrix as a spinor.

11.3.1 Screw and Twist

Having described finite motion in terms of motors to represent the Lie group of rigid motion $SE(3)$, the logical question of infinitesimal motions or the velocities arises. These are described by elements of the Lie algebra $se(3)$. There are two most useful interpretations to describe relation between $SE(3)$ and $se(3)$, one as the tangent vectors to the identity element or as the set of left-invariant vector fields on the group. We can see that these are equivalent since any left-invariant vector field determines a tangent vector at the identity. Furthermore, any tangent vector at the identity can be left translated over the group to produce a left-invariant vector field.

In motor algebra, we represent the rigid motion of lines by Eq. (4.67). Lines are represented in terms of two dual bivector basis. Following the same idea followed by the case of the pure rotor motion, we represent the relationship of the moving frames as follows:

$$\begin{aligned} \boldsymbol{L}' &= \boldsymbol{M}(t)\boldsymbol{L}\widetilde{\boldsymbol{M}}(t), \\ l'_k(t) &= \boldsymbol{M}(t) l_k \widetilde{\boldsymbol{M}}(t), \end{aligned} \tag{11.19}$$

where l'_k and l_k are the coefficients of the two frames. Taking its time derivative

$$\dot{l}'_k = \dot{\boldsymbol{M}} l_k \widetilde{\boldsymbol{M}} + \boldsymbol{M} l_k \dot{\widetilde{\boldsymbol{M}}}, \tag{11.20}$$

and substituting $\boldsymbol{M} l_k = l'_k \boldsymbol{M}$, one gets

$$\dot{l}'_k = \dot{\boldsymbol{M}} \widetilde{\boldsymbol{M}} l'_k + l'_k \boldsymbol{M} \dot{\widetilde{\boldsymbol{M}}}. \tag{11.21}$$

Taking the time derivative of the identity $\boldsymbol{M}\widetilde{\boldsymbol{M}} = 1$, we get the useful relations

$$\begin{aligned} \partial_t(\boldsymbol{M}\widetilde{\boldsymbol{M}}) &= 0, \\ \dot{\boldsymbol{M}}\widetilde{\boldsymbol{M}} + \boldsymbol{M}\dot{\widetilde{\boldsymbol{M}}} &= 0, \\ \dot{\boldsymbol{M}}\widetilde{\boldsymbol{M}} &= -\boldsymbol{M}\dot{\widetilde{\boldsymbol{M}}}, \end{aligned} \tag{11.22}$$

which are substituted in Eq. (11.21) to yield

$$\dot{l_k}'(t) = \dot{\boldsymbol{M}}\widetilde{\boldsymbol{M}}l_k' - l_k'\dot{\boldsymbol{M}}\widetilde{\boldsymbol{M}}. \tag{11.23}$$

Since the right side of the last equation is the inner product between a bivector and the vector basis coefficients, we rewrite it as follows:

$$\dot{l_k}' = (2\dot{\boldsymbol{M}}\widetilde{\boldsymbol{M}}) \cdot l_k'. \tag{11.24}$$

Let us call the bivector

$$\boldsymbol{s} = 2\dot{\boldsymbol{M}}\widetilde{\boldsymbol{M}} \tag{11.25}$$

the spatial velocity or twist; as we will show next it comprises a bivector for the angular velocity and a bivector for the linear velocity. Recalling that $\boldsymbol{M} = \boldsymbol{T}\boldsymbol{R}$, we proceed as follows:

$$\begin{aligned} \boldsymbol{s} = 2\dot{\boldsymbol{M}}\widetilde{\boldsymbol{M}} &= 2(\dot{\boldsymbol{T}}\boldsymbol{R} + \boldsymbol{T}\dot{\boldsymbol{R}})\widetilde{\boldsymbol{R}}\widetilde{\boldsymbol{T}} = 2\dot{\boldsymbol{T}}\boldsymbol{R}\widetilde{\boldsymbol{R}}\widetilde{\boldsymbol{T}} + 2\boldsymbol{T}\dot{\boldsymbol{R}}\widetilde{\boldsymbol{R}}\widetilde{\boldsymbol{T}}, \\ &= \left(\boldsymbol{\omega}_S + e_\infty \dot{\boldsymbol{t}}_S\right) = \left(\boldsymbol{\omega}_S + e_\infty \boldsymbol{v}_S\right), \end{aligned} \tag{11.26}$$

where the angular-velocity bivector $\boldsymbol{\omega}_S$ is the dual of the linear-velocity bivector $\boldsymbol{v}_S$.

Multiplying Eq. (11.25) from the right by $\boldsymbol{M}$ and dividing by -2, we get the dynamic motor equation

$$\dot{\boldsymbol{M}} = -\frac{1}{2}\boldsymbol{s}\boldsymbol{M}. \tag{11.27}$$

Assuming that the screw motion of the body is constant through time, the dynamic motor Eq. (11.27) can be integrated to give

$$\boldsymbol{M}(t) = e^{-\frac{\boldsymbol{s}}{2}}\boldsymbol{M}(0) = e^{-\frac{(\boldsymbol{\omega} + e_\infty \boldsymbol{v})}{2}}\boldsymbol{M}(0), \tag{11.28}$$

where $\boldsymbol{s} = \boldsymbol{\omega} + e_\infty \boldsymbol{v}$ stands for the twist. The screw motion is about and along the screw line $\boldsymbol{L}$, and hence the pitch can be computed as follows:

$$p = \frac{\boldsymbol{\omega} \cdot \boldsymbol{v}}{\boldsymbol{\omega} \cdot \boldsymbol{\omega}}. \tag{11.29}$$

The twist $\boldsymbol{s}$ is the representation of an element of the Lie algebra $se(3)$ in terms of a bivector basis which corresponds to the Lie algebra generators.

11.3.2 Momentum and Wrench

We will introduce the notion of a dual bivector with respect to a twist $\boldsymbol{s}$. The affine transformation of the inertia matrix $\mathtt{N}$ to the twist yields the momentum which written using 6×1 vectors reads

$$\begin{aligned} \boldsymbol{\wp} = \boldsymbol{j} + e_\infty \boldsymbol{p} = \mathtt{N}(\boldsymbol{s}) \equiv \mathtt{N}\mathbf{s} = \begin{bmatrix} \mathbf{j} \\ \mathbf{p} \end{bmatrix} = \begin{bmatrix} \mathtt{I} & m[\mathbf{c}]_x \\ m[\mathbf{c}]_x^T & mI \end{bmatrix} \begin{bmatrix} \omega \\ \mathbf{v} \end{bmatrix}, \\ \equiv \begin{bmatrix} \mathtt{I}\omega + m[\mathbf{c}]_x \mathbf{v} \\ m[\mathbf{c}]_x^T \omega + m\mathbf{v} \end{bmatrix}, \end{aligned} \tag{11.30}$$

where $\mathtt{I}$ is the inertia, m is the mass, and I is the identity matrix. Note that inertia matrix $\mathtt{N}$ acts as a dualizing matrix.

The kinetic energy of the body is given by inner product of the momentum co-twist and the twist

$$\begin{aligned} E_K &= \frac{1}{2}\boldsymbol{\wp} \cdot \boldsymbol{s} = \frac{1}{2}\mathtt{N}(\boldsymbol{s}) \cdot \boldsymbol{s} = \frac{1}{2}(\boldsymbol{j} \cdot \boldsymbol{w} + \boldsymbol{p} \cdot \boldsymbol{v}), \\ &\equiv \frac{1}{2}(\mathtt{N}\mathbf{s})^T \cdot \boldsymbol{s} = \frac{1}{2}\mathbf{s}^T \mathtt{N}\mathbf{s}. \end{aligned} \tag{11.31}$$

The time derivative of a twist $\boldsymbol{s}$ is the bivector acceleration $\dot{\boldsymbol{s}}$

$$\dot{\boldsymbol{s}} = \frac{d\boldsymbol{s}}{dt} = \dot{\boldsymbol{\omega}} + \epsilon_\infty \dot{\boldsymbol{v}} \equiv \begin{bmatrix} \frac{d\boldsymbol{\omega}}{dt} \\ \frac{d\boldsymbol{v}}{dt} \end{bmatrix} = \begin{bmatrix} \dot{\boldsymbol{\omega}} \\ \dot{\boldsymbol{v}} \end{bmatrix} \tag{11.32}$$

and for $\mathtt{N}$ time-invariant, the time derivative of a momentum $\boldsymbol{\wp}$ is the wrench $\boldsymbol{w}$, namely,

$$\begin{aligned} \boldsymbol{w} &= \boldsymbol{\tau} + e_\infty \boldsymbol{f}, \\ &= \dot{\boldsymbol{\wp}} = \frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{j} + e_\infty \frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{p} = \frac{\mathrm{d}}{\mathrm{d}t}(\mathtt{I}(\boldsymbol{\omega}) + m\boldsymbol{c} \wedge \boldsymbol{v}) + e_\infty \frac{\mathrm{d}}{\mathrm{d}t}(m\boldsymbol{\omega} \wedge \boldsymbol{c} + m\boldsymbol{v}), \\ &= \mathtt{N}(\dot{\boldsymbol{s}}) = \mathtt{N}(\frac{d\boldsymbol{s}}{dt}) = \mathtt{N}(\dot{\boldsymbol{\omega}} + \epsilon_\infty \dot{\boldsymbol{v}}), \end{aligned} \tag{11.33}$$

where $\boldsymbol{\tau}$ and $\boldsymbol{f}$ stand for the bivectors torque and force, respectively. This equation of the wrench written using 6×1 vectors reads

$$\mathbf{w} = \begin{bmatrix} \tau \\ f \end{bmatrix} = \mathbb{N}(\begin{bmatrix} \frac{d\omega}{dt} \\ \frac{d\mathbf{v}}{dt} \end{bmatrix}) = \begin{bmatrix} \mathtt{I} & m[\mathbf{c}]_x \\ m[\mathbf{c}]_x^T & mI \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{\omega}} \\ \dot{\mathbf{v}} \end{bmatrix},$$
$$= \begin{bmatrix} \mathtt{I}\dot{\boldsymbol{\omega}} + m[\mathtt{t}]_\times \dot{\mathbf{v}} \\ m[\mathtt{t}]_\times^T \dot{\boldsymbol{\omega}} + mI\dot{\mathbf{v}} \end{bmatrix}. \tag{11.34}$$

The inner product between the wrench and the twist is a scalar

$$\boldsymbol{w} \cdot \boldsymbol{s} = \boldsymbol{\tau} \cdot \boldsymbol{\omega} + \boldsymbol{f} \cdot \boldsymbol{v} \equiv \mathbb{N}(\dot{\boldsymbol{s}}) \cdot \boldsymbol{s} = \frac{1}{2}\dot{\mathbf{s}}^T \mathbb{N}\mathbf{s}. \tag{11.35}$$

The significance of the wrench is that they can be used to represent the momentum of a rigid body, and similarly, they can describe the action of a wrench on a rigid body, that is, a torque/force bivector.

Finally, the action of the group on the twist and wrenches using motors is given by

$$\boldsymbol{s}' = \boldsymbol{M}\boldsymbol{s}\widetilde{\boldsymbol{M}}, \tag{11.36}$$
$$\boldsymbol{w}' = \boldsymbol{M}\boldsymbol{w}\widetilde{\boldsymbol{M}}. \tag{11.37}$$

Consider the twist $\boldsymbol{s}_j$ attached to a j-joint of a moving rigid body; as a function of the time, the equation of an infinitesimal motion in terms of an instantaneous velocity twist $\boldsymbol{s}_i$ acting on a twist $\boldsymbol{s}_j$ is given by

$$\boldsymbol{s}_j(t) = \boldsymbol{M}(t)\boldsymbol{s}_j(0)\widetilde{\boldsymbol{M}}(t) = e^{\frac{t}{2}\boldsymbol{s}_i}\boldsymbol{s}_j(0)e^{-\frac{t}{2}\boldsymbol{s}_i}, \tag{11.38}$$

where $\boldsymbol{s}_i = \boldsymbol{\omega}_i + e_\infty \boldsymbol{v}_i$ and $\boldsymbol{s}_j = \boldsymbol{\omega}_j + e_\infty \boldsymbol{v}_j$. Differentiating and setting $t = 0$, we get the action of $\boldsymbol{s}_i$ on $\boldsymbol{s}_j$, namely,

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{s}_j(t) &= (\frac{\mathrm{d}}{\mathrm{d}t}e^{\frac{t}{2}\boldsymbol{s}_i}\boldsymbol{s}_j(0)e^{-\frac{t}{2}\boldsymbol{s}_i})|_{t=0}, \\ &= (\frac{1}{2}\boldsymbol{s}_i e^{\frac{t}{2}\boldsymbol{s}_i}\boldsymbol{s}_j(0)e^{-\frac{t}{2}\boldsymbol{s}_i} - \frac{1}{2}e^{\frac{t}{2}\boldsymbol{s}_i}\boldsymbol{s}_j(0)e^{-\frac{t}{2}\boldsymbol{s}_i}\boldsymbol{s}_i)|_{t=0}, \\ &= \frac{1}{2}(\boldsymbol{s}_i\boldsymbol{s}_j - \boldsymbol{s}_j\boldsymbol{s}_i) = [\boldsymbol{s}_i, \boldsymbol{s}_j], \end{aligned} \tag{11.39}$$

where $[\boldsymbol{s}_i, \boldsymbol{s}_j]$ is known as the Lie bracket of $\boldsymbol{s}_i$ and $\boldsymbol{s}_j$. Since this result does not depend on where the measurement time begins, in general, for two instantaneous twists $\boldsymbol{s}_i$ and $\boldsymbol{s}_j$, we can claim

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{s}_j(t) &= [\boldsymbol{s}_i, \boldsymbol{s}_j], \\ &= [(\boldsymbol{\omega}_i + e_\infty \boldsymbol{v}_i), (\boldsymbol{\omega}_j + e_\infty \boldsymbol{v}_j)] \equiv \begin{bmatrix} \mathbf{w}_i \times \mathbf{w}_j \\ \mathbf{v_i} \times \mathbf{w_j} + \mathbf{w_i} \times \mathbf{v_j} \end{bmatrix}. \end{aligned} \tag{11.40}$$

Now consider the wrench $\boldsymbol{w}_j$ attached to a j-joint of a moving rigid body; as a function of the time, the equation of an infinitesimal motion in terms of an instantaneous velocity twist $\boldsymbol{s}_i$ acting on a wrench $\boldsymbol{w}_j$ is given by

$$\boldsymbol{w}_j(t) = \widetilde{\boldsymbol{M}}(t)\boldsymbol{w}_j(0)\boldsymbol{M}(t) = e^{-\frac{t}{2}\boldsymbol{s}_i}\boldsymbol{w}_j(0)e^{\frac{t}{2}\boldsymbol{s}_i}, \tag{11.41}$$

where $\boldsymbol{s}_i = \boldsymbol{\omega}_i + e_\infty \boldsymbol{v}_i$ and $\boldsymbol{w}_j = \boldsymbol{\tau}_j + e_\infty \boldsymbol{f}_j$. Differentiating and setting $t = 0$, we get the action of $\boldsymbol{s}_i$ on $\boldsymbol{w}_j$, namely,

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{w}_j(t) &= (\frac{\mathrm{d}}{\mathrm{d}t}e^{-\frac{t}{2}\boldsymbol{s}_i}\boldsymbol{w}_j(0)e^{\frac{t}{2}\boldsymbol{s}_i})|_{t=0},\\ &= (-\frac{1}{2}\boldsymbol{s}_i e^{-\frac{t}{2}\boldsymbol{s}_i}\boldsymbol{w}_j(0)e^{\frac{t}{2}\boldsymbol{s}_i} + \frac{1}{2}e^{-\frac{t}{2}\boldsymbol{s}_i}\boldsymbol{w}_j(0)e^{\frac{t}{2}\boldsymbol{s}_i}\boldsymbol{s}_i)|_{t=0},\\ &= \frac{1}{2}(-\boldsymbol{s}_i\boldsymbol{w}_j + \boldsymbol{w}_j\boldsymbol{s}_i) = \{\boldsymbol{s}_i, \boldsymbol{w}_j\},\end{aligned} \tag{11.42}$$

where $\{\boldsymbol{s}_i, \boldsymbol{w}_j\}$ is known as the Lie co-bracket of $\boldsymbol{s}_i$ and $\boldsymbol{w}_j$. Since this result does not depend on where the measurement time begins, in general, for instantaneous twist $\boldsymbol{s}_i$ and wrench $\boldsymbol{w}_j$, we can claim

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{w}_j(t) &= \{\boldsymbol{s}_i, \boldsymbol{w}_j\},\\ &= \{(\boldsymbol{\omega}_i + e_\infty\boldsymbol{v}_i), (\boldsymbol{\tau}_j + e_\infty\boldsymbol{f}_j)\} \equiv \begin{bmatrix}\mathbf{w}_i \times \tau_j + \mathbf{v_i} \times \mathbf{f_j}\\ \omega_i \times \mathbf{f}_j\end{bmatrix}.\end{aligned} \tag{11.43}$$

11.3.3 Compliance

In the practice, all robot joints and links are slighter complaint, but the joint compliance is normally much greater than the link compliance. The physical property of springiness in the joints is mainly due to the springiness of gears and shafts; however, there is also some extra springiness which comes from the control system and the electric motors as well. Considering a simple model for this effect, a deflection $\delta\theta_i$ in the i-th joint will cause a torque $\tau_i = k_i\delta\theta_i$, where k_i is the joint stiffness. In the next subsections, we will use bold fonts for vectors and slant vectors for bivectors and matrices for linear transformation of vectors.

For a serial manipulator, this can be formulated using a matrix as follows:

$$\tau = K\triangle\theta, \tag{11.44}$$

where K is a diagonal matrix of joint stiffness and is called stiffness matrix of the robot. For a n-D.O.F robot manipulator in equilibrium, the wrench at the end-effector $\boldsymbol{w}$, resulting from the torques at the joints, is given as a vector by

$$\mathbf{w} = (J^T)^{-1}\tau, \tag{11.45}$$

where J is the manipulator Jacobian. The displacement of the end-effector in terms of the joint displacement is

$$\triangle \mathtt{x} = J \triangle \theta. \tag{11.46}$$

Combining these equations

$$\mathbf{w} = (J^T)^{-1}\tau = (J^T)^{-1}\mathtt{K}\triangle\theta = (J^T)^{-1}\mathtt{K}J^{-1}\triangle\mathbf{x}. \tag{11.47}$$

This equation represents the wrench caused by a small deflection in the position of the end-effector. It is often used as the compliance matrix $\mathtt{C}$ of the manipulator which is defined as $\mathtt{C} = J\mathtt{K}^{-1}J^T$, and thus the wrench reads

$$\mathbf{w} = \mathtt{C}^{-1}\triangle\mathbf{x}. \tag{11.48}$$

If the Jacobian becomes singular the compliance matrix C is then undefined, which means that the arm is infinitely stiff in at least one direction.

11.3.4 Moments of Inertia

The lengths of the rectangular box of Fig. 11.1 are 2a, 2b, and 2c, and they are aligned w.r.t. the $x-$, $y-$, and z-axes, respectively, and the origin is located at the center of the block. If the block has an uniform density ρ, its total mass is computed as follows:

$$m = 8\rho abc. \tag{11.49}$$

The first term of the inertia matrix of Eq. (11.17) is computed as

$$\begin{aligned}\int \rho(y^2+z^2)dvol &= \rho\int_{-a}^{+a}\mathtt{d}x\left(\int_{-b}^{+b}y^2\mathtt{d}y\int_{-c}^{+c}\mathtt{d}z + \int_{-b}^{+b}\mathtt{d}y\int_{-c}^{+c}z^2\mathtt{d}z\right),\\ &= 8\rho a(\frac{b^3}{3}c + b\frac{c^3}{3}) = \frac{m}{3}(b^2+c^2).\end{aligned} \tag{11.50}$$

The other diagonal terms are similar and they can be easily computed by relabeling the variables. The first off-diagonal element in the matrix is given by

$$\int \rho xy\mathtt{d}vol = \rho\int_{-a}^{+a}x\mathtt{d}x\int_{-b}^{+b}y\mathtt{d}y\int_{-c}^{+c}\mathtt{d}z, \tag{11.51}$$

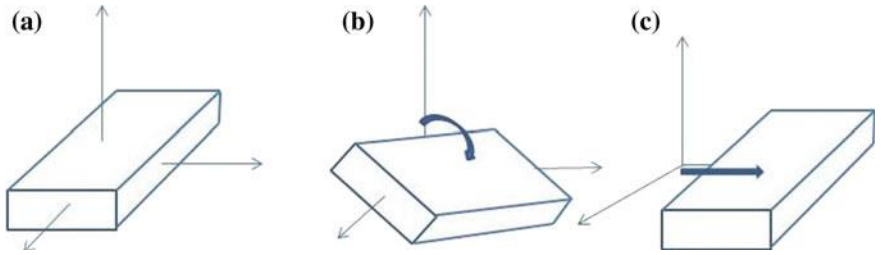


Fig. 11.1 **a** A rectangular box. **b** After a rotation. **c** After a translation

and the first two integrals in the product are zero. Since the off-diagonal entries are similar, we relabel them and they vanish as well. Thus, the 3×3 inertia matrix reads

$$\mathtt{I} = \begin{bmatrix} \frac{m}{3}(b^2+c^2) & 0 & 0 \\ 0 & \frac{m}{3}(a^2+c^2) & 0 \\ 0 & 0 & \frac{m}{3}(a^2+b^2) \end{bmatrix}. \tag{11.52}$$

11.3.5 Rigid Transformation of the Inertia

The inertia was computed assuming that its origin and axes are located on the coordinate system of $\mathbb{R}^3$. If a rotation acts on the inertia $\mathtt{I}$, the integrals have to involve the rotation $\mathtt{R}$. The inertia is defined as

$$\mathtt{I}\mathbf{w} = \int \rho\mathbf{x} \times (\mathbf{w} \times \mathbf{x})\mathrm{d}vol,$$

then the rotated inertia reads

$$\mathtt{I}'\mathbf{w} = \int \rho\mathtt{R}\mathbf{x} \times (\mathbf{w} \times \mathtt{R}\mathbf{x})\mathrm{d}vol, \tag{11.53}$$

since $\mathtt{R}(\mathbf{p} \times \mathbf{q}) = \mathtt{R}\mathbf{p} \times \mathtt{R}\mathbf{q}$, then

$$\mathtt{I}'\mathbf{w} = \mathtt{R}\int \rho\mathbf{x}(\mathtt{R}^{-1}\mathbf{w} \times \mathbf{x})\mathrm{d}vol = \mathtt{R}\mathtt{N}\mathtt{R}^{-1}\mathbf{w}, \tag{11.54}$$

thus the inertia is given by the similarity transformation,

$$\mathtt{I}' = \mathtt{R}\mathtt{N}\mathtt{R}^{-1}. \tag{11.55}$$

If we rotate the inertia with respect to the z-axis, the inertia is

$$
\begin{aligned}
&\mathtt{I}' = \mathtt{R}\mathtt{N}\mathtt{R}^{-1},\\
&= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) \\ 0 & \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} \frac{m}{3}(b^2+c^2) & 0 & 0 \\ 0 & \frac{m}{3}(a^2+c^2) & 0 \\ 0 & 0 & \frac{m}{3}(a^2+b^2) \end{bmatrix}\\
&\begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & \sin(\theta) \\ 0 & -\sin(\theta) & \cos(\theta) \end{bmatrix},\\
&= \frac{m}{3} \begin{bmatrix} (b^2+c^2) & 0 & 0 \\ 0 & (a^2+b^2\sin^2\theta + c^2\cos^2\theta) & 0 \\ 0 & (c^2-b^2)\cos\theta\sin\theta & a^2+b^2\cos^2\theta + c^2\sin^2\theta \end{bmatrix}.
\end{aligned}
\tag{11.56}
$$

The translation of the inertia is computed shifting in $\boldsymbol{t}$ the point $\boldsymbol{x}$ as follows:

$$
\mathtt{I}''\mathbf{w} = \int \rho(\mathbf{x}+\mathbf{t}) \times (\mathbf{w} \wedge (\mathbf{x}+\mathbf{t}))\mathtt{d}vol, \tag{11.57}
$$

since the wedge is distributive w.r.t. the wedge product,

$$
\begin{aligned}
(\mathbf{x}+\mathbf{t}) \ &\times(\mathbf{w} \times (\mathbf{x}+\mathbf{t})) = \\
&\mathbf{x} \times (\mathbf{w} \times \mathbf{x}) + \mathbf{t} \times (\mathbf{w} \times \mathbf{x}) + \mathbf{x} \times (\mathbf{w} \times \mathbf{t}) + \mathbf{t} \times (\mathbf{w} \times \mathbf{t}).
\end{aligned}
$$

Since

$$
\int \rho \boldsymbol{x} \mathtt{d}vol = m\boldsymbol{c}, \tag{11.58}
$$

and computing the integral of the expansion, we can compute the translated inertia as follows:

$$
\mathtt{I}''\mathbf{w} = \mathtt{I}\mathbf{w} + m\mathbf{t} \times (\mathbf{w} \times \mathbf{c}) + m\mathbf{c} \times (\mathbf{w} \times \mathbf{t}) + m\mathbf{t} \times (\mathbf{w} \times \mathbf{t}), \tag{11.59}
$$

where $\boldsymbol{c}$ is the body's center of mass in the original position. Now, using vector representation and the antisymmetric matrices $\mathtt{T} = [\mathbf{t}]_x$ and $\mathtt{C} = [\mathbf{c}]_x$ representing the cross products with $\boldsymbol{t}$ and $\boldsymbol{c}$, respectively, the translated inertia matrix reads

$$
\mathtt{I}''\mathbf{w} = (\mathtt{I} - m\mathtt{T}\mathtt{C} - m\mathtt{C}\mathtt{T} - m\mathtt{T}^2)\mathbf{w}, \tag{11.60}
$$

thus the inertia matrix is

$$
\mathtt{I}'' = (\mathtt{I} - m\mathtt{T}\mathtt{C} - m\mathtt{C}\mathtt{T} - m\mathtt{T}^2). \tag{11.61}
$$

This result is called the parallel axis theorem. Let us compute the inertia translated in $\mathbf{t} = [t_x, t_y, t_z]^T$ for an inertia located at the origin, thus $\mathtt{C} = \mathbf{0}$

$$
\begin{aligned}
\mathtt{I}'' &= (\mathtt{I} - m\mathtt{T}\mathtt{C} - m\mathtt{C}\mathtt{T} - m\mathtt{T}^2) = (\mathtt{I} - m\mathtt{T}\mathbf{0} - m\mathbf{0}\mathtt{T} - m\mathtt{T}^2) = (\mathtt{I} - m\mathtt{T}^2),\\
&= \begin{bmatrix} \frac{m}{3}(b^2+c^2) & 0 & 0 \\ 0 & \frac{m}{3}(a^2+c^2) & 0 \\ 0 & 0 & \frac{m}{3}(a^2+b^2) \end{bmatrix},\\
&- m \begin{bmatrix} -(t_y^2+t_z^2) & t_x t_y & t_x t_z \\ t_x t_y & -(t_x^2+t_z^2) & t_y t_z \\ t_x t_z & t_y t_z & -(t_x^2+t_y^2) \end{bmatrix},\\
&= m \begin{bmatrix} \frac{1}{3}(b^2+c^2)+(t_y^2+t_z^2) & -t_x t_y & -t_x t_z \\ -t_x t_y & \frac{1}{3}(a^2+c^2)+(t_x^2+t_y^2) & -t_y t_z \\ -t_x t_z & -t_y t_z & \frac{1}{3}(a^2+b^2)+(t_x^2+t_y^2) \end{bmatrix}.
\end{aligned}
$$

Screw lines $\boldsymbol{L} = \boldsymbol{n} + e_\infty \boldsymbol{m}$ transform under the action of $SE(3)$ or $\boldsymbol{M}$ (motor) as follows:

$$\boldsymbol{L}' = \boldsymbol{M}\boldsymbol{L}\widetilde{\boldsymbol{M}}, \tag{11.62}$$

in which using a representation in terms of a vector and an homogeneous transformation matrix, this equation reads

$$\mathtt{L}' = \mathtt{M}\begin{bmatrix} \mathbf{n} \\ \mathbf{m} \end{bmatrix} = \begin{bmatrix} \mathtt{R} & \mathtt{O} \\ [\mathbf{t}]_x \mathtt{R} & \mathtt{R} \end{bmatrix}\begin{bmatrix} \mathbf{n} \\ \mathbf{m} \end{bmatrix}. \tag{11.63}$$

Since in motor algebra $G^+_{3,0,1}$ or conformal geometric algebra $G_{4,1}$, we can't represent the inertia matrix $\mathtt{N}$ as spinor, we have to formulate its sandwich transformation using homogeneous matrices as follows, since

$$\mathtt{M}_i = \begin{bmatrix} \mathtt{R} & \mathtt{O} \\ [\mathbf{t}]_x \mathtt{R} & \mathtt{R} \end{bmatrix},\ (\mathtt{M}^T)^{-1} = \begin{bmatrix} \mathtt{R} & [\mathbf{t}]_x \mathtt{R} \\ \mathtt{O} & \mathtt{R} \end{bmatrix},\ \mathtt{M}^{-1} = \begin{bmatrix} \mathtt{R}^T & \mathtt{O} \\ -\mathtt{R}^T [\mathbf{t}]_x & \mathtt{R}^T \end{bmatrix}, \tag{11.64}$$

so under the rotation and translation the 6×6 inertia matrix $\mathtt{N}$ is transformed as

$$\mathtt{N}' = (\mathtt{M}^T)^{-1}\mathtt{N}\mathtt{M}^{-1}. \tag{11.65}$$

Note that this relation is for a rotation followed by a translation, thus the new center of mass is given by

$$[\mathbf{c}]'_x = \mathtt{R}[\mathbf{c}]_x\mathtt{R}^T + [\mathbf{t}]_x. \tag{11.66}$$

11.3.6 Time Derivatives of the Inertia

If we consider a particle with constant mass, the linear momentum is independent of the mass and the equation of the motion is $\mathbf{F} = m\dot{\mathbf{v}}$. Similarly considering that

the body has a constant mass and is not deformable, the equation of motion of the inertia at t=0 is $\mathtt{I}_0$ and after subsequent times if the body undergoes a rotation about the origin, the inertia matrix is given by

$$\mathtt{I} = \mathtt{R}\mathtt{I}_0\mathtt{R}^T, \tag{11.67}$$

where the rotation matrix $\mathtt{R}$ is a function of time, differentiating the inertia with respect to time we get

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathtt{I} = \dot{\mathtt{R}}\mathtt{I}_0\mathtt{R}^T + \mathtt{R}\mathtt{I}_0\dot{\mathtt{R}}^T. \tag{11.68}$$

The angular momentum is given by $\mathbf{J} = \mathtt{I}\mathbf{w}$, and its time derivative gives

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathtt{J} = \mathtt{I}\dot{\mathbf{w}} + \dot{\mathtt{R}}\mathtt{I}_0\mathtt{R}^T\mathbf{w} + \mathtt{R}\mathtt{I}_0\dot{\mathtt{R}}^T\mathbf{w}, \tag{11.69}$$

since $\dot{\mathtt{R}}^T\mathbf{w} = -\mathbf{w} \times \mathbf{w} = 0$ and at $t = 0$ $\mathtt{R}(0) = I$, the derivative of the angular momentum is therefore

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathtt{J}(0) = \mathtt{I}_0(\dot{\mathbf{w}}) + \mathbf{w} \times \mathtt{I}_0\mathtt{R}(0)^T\mathbf{w} + \mathtt{R}\mathtt{I}_0 - \mathbf{w} \times \mathbf{w} = \mathtt{I}_0\dot{\mathbf{w}} + \mathbf{w} \times \mathtt{I}_0\mathbf{w}. \tag{11.70}$$

Since the time begins in any instant, the previous equation can be generally true and it equals to the torque applied to the rigid body

$$\tau = \frac{\mathrm{d}}{\mathrm{d}t}\mathtt{J} = \mathtt{I}\dot{\mathbf{w}} + \mathbf{w} \times \mathtt{I}\mathbf{w}. \tag{11.71}$$

This equation is known as Euler's equation which is written normally in terms of coordinates at rest with respect to the body, so the 3×3 inertia matrix is constant. However, one uses a single, fixed coordinate frame, so the inertia matrix must vary and the equation is exactly the same.

For the case of a general rigid motion, we can extend the derivation given above using the 6×6 inertia matrix $\mathtt{N}$. As we said before, since we have not a spinor for the $\mathtt{N}$, we have to derive the equations using the $SE(3)$ transformations as follows:

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathtt{M} = \frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix} \mathtt{R} & \mathtt{O} \\ [\mathbf{t}]_x\,\mathtt{R} & \mathtt{R} \end{bmatrix} = \begin{bmatrix} \dot{\mathtt{R}} & \mathtt{O} \\ [\dot{\mathbf{t}}]_x\,\mathtt{R} + [\mathbf{t}]_x\,\dot{\mathtt{R}} & \dot{\mathtt{R}} \end{bmatrix}. \tag{11.72}$$

Now, when $t = 0$ the action of $SE(3)$ using a 6×6 matrix $\mathtt{M}$ on an arbitrary instantaneous twist $\boldsymbol{s}_j = \boldsymbol{w}_j + e_\infty \boldsymbol{v}_j$, we can rewrite Eq. (11.40) in terms of matrices and 6D vectors as follows:

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathtt{M}(0)\begin{bmatrix} \mathbf{w} \\ \mathbf{v} \end{bmatrix} = \begin{bmatrix} \dot{\mathtt{R}} & \mathtt{O} \\ [\dot{\mathbf{t}}]_x\,\mathtt{R} + [\mathbf{t}]_x\,\dot{\mathtt{R}} & \dot{\mathtt{R}} \end{bmatrix}\begin{bmatrix} \mathbf{w} \\ \mathbf{v} \end{bmatrix} = \begin{bmatrix} \mathbf{w}_i \times \mathbf{w}_j \\ \mathbf{v}_i \times \mathbf{w}_j + \mathbf{w}_i \times \mathbf{v}_j \end{bmatrix},$$

since this result does not depend on where the measurement time begins, in general for two instantaneous twists, we compute their Lie bracket as follows:

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathtt{M}_i \begin{bmatrix} \mathbf{w}_j \\ \mathbf{v}_j \end{bmatrix} = [\mathbf{s}_i, \mathbf{s}_j] = \begin{bmatrix} \mathbf{w}_i \times \mathbf{w}_j \\ \mathbf{v}_i \times \mathbf{w}_j + \mathbf{w}_i \times \mathbf{v}_j \end{bmatrix}. \tag{11.73}$$

In this regard, using similar arguments, we can compute the time derivative of the matrix $(\mathtt{M}^T)^{-1}$ acting on an arbitrary wrench $\boldsymbol{w}_j = \boldsymbol{w}_j + e_\infty \boldsymbol{v}_j$, and we can rewrite Eq. (11.43) in terms of matrices and 6D vectors as follows:

$$\frac{\mathrm{d}}{\mathrm{d}t}(\mathtt{M}^T)^{-1}(0) \begin{bmatrix} \tau \\ \mathbf{f} \end{bmatrix} = \begin{bmatrix} \dot{\mathtt{R}} & [\dot{\mathbf{t}}]_x \mathtt{R} + [\mathbf{t}]_x \dot{\mathtt{R}} \\ \mathtt{O} & \dot{\mathtt{R}} \end{bmatrix} \begin{bmatrix} \tau \\ \mathbf{f} \end{bmatrix} = \{\mathbf{s}_i, \mathbf{w}_j\}. \tag{11.74}$$

Since this result does not depend on where the measurement time begins, in general for instantaneous twist and wrench, we compute their Lie co-bracket as follows:

$$\frac{\mathrm{d}}{\mathrm{d}t}(\mathtt{M}^T)^{-1} \begin{bmatrix} \tau_j \\ \mathbf{f}_j \end{bmatrix} = \{\mathbf{s}_i, \mathbf{w}_j\} = \begin{bmatrix} \mathbf{w}_i \times \tau_j + \mathbf{v}_i \times \mathbf{f}_j \\ \mathbf{w}_i \times \mathbf{f}_j \end{bmatrix}. \tag{11.75}$$

Now, it is a straightforward task to derive the time derivative of the 6×6 inertia matrix. The action of an infinitesimal motion of the $SE(3)$ Lie group on the inertia matrix is given by

$$\mathtt{N}(t) = e^{-[\mathbf{S}]_\times} \mathtt{N}(0) e^{[\mathbf{S}]_\times}, \tag{11.76}$$

where $\mathbf{S}$ is the 6×1 vector representation of the twist $\boldsymbol{s}$, taking the derivative

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t}\mathtt{N}(t) &= -adj(\mathbf{S})^T e^{-[\mathbf{S}]_\times} \mathtt{N}(0) e^{[\mathbf{S}]_\times} + e^{-[\mathbf{S}]_\times} \mathtt{N}(0) e^{[\mathbf{S}]_\times} adj(\mathbf{S}), \\ &= -adj(\mathbf{S})^T \mathtt{N}(t) + \mathtt{N}(t) adj(\mathbf{S}) = \{\mathbf{S}, \mathtt{N}\mathbf{S}\}, \end{aligned} \tag{11.77}$$

since this result does not depend on where the measurement time begins, in general for instantaneous velocity or twist and inertia matrix $\mathtt{N}(t)$, we compute their Lie co-bracket as follows:

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathtt{N}(t) = \{\mathbf{S}, \mathtt{N}\mathbf{S}\} \equiv \{\boldsymbol{s}, \mathtt{N}(\boldsymbol{s})\}. \tag{11.78}$$

The rate of change of momentum equals to the wrench $\boldsymbol{w}$

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathtt{N}(t)\boldsymbol{s} = \boldsymbol{w}. \tag{11.79}$$

Evaluating the derivative of the inertia matrix, we obtain

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathtt{N}(t)\boldsymbol{s} = \mathtt{N}\dot{\boldsymbol{s}} - (\boldsymbol{s}(e^{-s}\mathtt{N}(0)e^{-s}) + (e^{-s}\mathtt{N}(0)e^{-s})\boldsymbol{s})\boldsymbol{s} = \mathtt{N}\dot{\boldsymbol{s}} + \{\boldsymbol{s}, \mathtt{N}\boldsymbol{s}\}. \quad (11.80)$$

Finally, we can write the equation of motion of a single rigid body as follows:

$$\boldsymbol{w} = \mathtt{N}\dot{\boldsymbol{s}} + \{\boldsymbol{s}, \mathtt{N}\boldsymbol{s}\}. \quad (11.81)$$

Note that this equation is not quite the screw combination of Newton's and Euler's equation, it embodies Newton's equation as the description of the velocity of the body's center of mass, unlike Euler's equation, the inertia matrix is not constant in the above equation, but changes as the inertia changes while the body moves. In order to formulate the equations for a single rigid body, one should fix the coordinate frame in the body and use the Coriolis theorem to derive the derivatives with respect to the selected frame, see [[4], Sects. 1.2, 3.2]. Equation (11.81) of motion of a single rigid body written in terms of a 6×6 inertia matrix and 6D vectors is given by

$$\begin{bmatrix}\tau_{tot}\\ \mathbf{f}_{tot}\end{bmatrix} = \begin{bmatrix}\mathtt{I} & m[\mathbf{c}]_x\\ m[\mathbf{c}]_x^T & mI\end{bmatrix}\begin{bmatrix}\dot{\mathbf{w}}\\ \dot{\mathbf{v}}\end{bmatrix} + \left\{\begin{bmatrix}\mathbf{w}\\ \mathbf{v}\end{bmatrix}, \begin{bmatrix}\mathtt{I} & m[\mathbf{c}]_x\\ m[\mathbf{c}]_x^T & mI\end{bmatrix}\begin{bmatrix}\mathbf{w}\\ \mathbf{v}\end{bmatrix}\right\}, \quad (11.82)$$

which after few simplifications yields

$$\tau_{tot} = \mathtt{I}\dot{\mathbf{w}} + \mathbf{w} \times (\mathtt{I}\mathbf{w}) + m\mathbf{c} \times (\dot{\mathbf{v}} + \mathbf{w} \times \mathbf{v}), \quad (11.83)$$

$$\mathbf{f}_{tot} = m(\dot{\mathbf{v}} + (\dot{\mathbf{w}} - \mathbf{v}) \times \mathbf{c}). \quad (11.84)$$

11.4 Newton–Euler Recursive Algorithm for Serial Robot Manipulators

Robot serial manipulators of n-D degrees of freedom can have revoluted, prismatic, or helical joints. In order to introduce the theme, we will start with just a revoluted single driven link as shown in Fig. 11.2.

11.4.1 Single Robot Link

The total wrench acting on a body has three sources: the torque τ due to the join motor, the weight of the link $\boldsymbol{G}$, and the reaction wrench $\boldsymbol{R}$ at the bearing, thus the total wrench is given by

$$\boldsymbol{W} = \boldsymbol{w} + \boldsymbol{G} + \boldsymbol{R}, \quad (11.85)$$

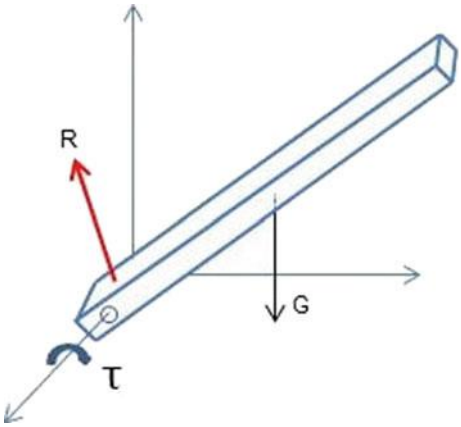


Fig. 11.2 A robot arm with one revoluted driven link

where $\boldsymbol{w} = \tau e_{23}$, τ is the magnitude of the torque and the weight $\boldsymbol{G}$ reads

$$\tau = \begin{bmatrix} \tau \\ 0 \\ 0 \\ \mathbf{0} \end{bmatrix}, \quad \boldsymbol{G} = \mathbb{N}(-ge_{\infty 3}) \equiv \begin{bmatrix} \mathbb{I} & m[\mathbf{c}]_x \\ m[\mathbf{c}]_x^T & mI \end{bmatrix} \begin{bmatrix} \mathbf{0} \\ 0 \\ 0 \\ g \end{bmatrix}, \tag{11.86}$$

$$\boldsymbol{G} = -mg\boldsymbol{c} \wedge e_{\infty 3} - mge_{\infty 3} \equiv \begin{bmatrix} -mg\boldsymbol{c} \wedge \boldsymbol{e}_{\infty 3} \\ 0 \\ 0 \\ -mge_{\infty 3} \end{bmatrix},$$

where g is the acceleration due to the gravity and m is the mass of the link. If we compute the inner product between Eq. (11.81) of motion for a rigid body with the screw line $\boldsymbol{L} = e_{23}$ passing through its joint, we get

$$\begin{aligned} \boldsymbol{W} \cdot \boldsymbol{L} &= (\mathbb{N}(\dot{\boldsymbol{s}}) + \{\boldsymbol{s}, \mathbb{N}(\boldsymbol{s})\}) \cdot \boldsymbol{L}, \\ &= \boldsymbol{w} - mg(\boldsymbol{c} \wedge e_{12}) \cdot e_{23} = \tau - mge_{31} \cdot \boldsymbol{c}. \end{aligned} \tag{11.87}$$

Since the body rotate about its joint screw line, the velocity screw or twist of the body is

$$\boldsymbol{s} = \dot{\theta}\boldsymbol{L}, \tag{11.88}$$

where θ is the joint angle, thus the Newton–Euler equation becomes

$$\boldsymbol{W} \cdot \boldsymbol{L} \equiv \mathbf{L}^T \mathbb{N}\mathbf{L}\ddot{\theta} + \{\mathbf{L}, \mathbb{N}\mathbf{L}\}^T \mathbf{L}\dot{\theta}^2. \tag{11.89}$$

Similarly as Eq. (11.83), we can write

$$\tau = e_{23} \cdot \mathtt{I}(e_{23}) + mge_{31} \cdot \boldsymbol{c}. \tag{11.90}$$

Finally, one has to consider that the center of mass and the inertia are position dependent, so in the initial position, where $\theta = 0$ the center of mass and the inertia will be $\boldsymbol{c}(0)$ and $\mathtt{I}(0)$, respectively. For sake of simplicity, consider that the center of mass is $\boldsymbol{c}(0) = c_y(0)e_2$. For a rotation about the z-axis,

$$\mathtt{R} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) \\ 0 & \sin(\theta) & \cos(\theta) \end{bmatrix}, \tag{11.91}$$

one computes

$$\mathtt{I}(\theta) = \mathtt{R}\mathtt{I}(0)\mathtt{R}^T \text{ and } \boldsymbol{c}(\theta) = \mathtt{R}\boldsymbol{c}(0), \tag{11.92}$$

and the resulting equation of motion reads

$$\tau = \mathtt{I}_{xx}(0) + mgc_y(0)\cos(\theta). \tag{11.93}$$

Finally, we compute the reaction force in the z-direction R_z. Using the Newton–Euler equation (11.85) and pairing with the line screw $\boldsymbol{Q} = e_{\infty 3}$

$$R_z = \boldsymbol{W} \cdot \boldsymbol{Q} \equiv \mathbf{L}^T \mathtt{N}\mathbf{Q}\ddot{\theta} + \{\mathbf{L}, \mathtt{N}\mathbf{L}\}^T \mathbf{Q}\dot{\theta}^2 + \mathbf{W} \cdot \mathbf{Q}, \tag{11.94}$$

which yields

$$R_z = -mc_y\ddot{\theta} + mc_z\dot{\theta}^2 - Mg. \tag{11.95}$$

If we consider that the center of mass lies along the y-axis, this equation is given by

$$R_z = -mc_y(0)\cos(\theta)\ddot{\theta} - mc_y(0)\sin(\theta)\dot{\theta}^2 - Mg. \tag{11.96}$$

11.4.2 Two Robot Links

We extend the previous formulas to the case of two robot links, as depicted in Fig. 11.3. Since there are two links, the model requires two Newton–Euler equations as follows:

$$\begin{aligned} \boldsymbol{w}_2 + \boldsymbol{R}_2 + \boldsymbol{G}_2 &= \mathtt{N}_2(\dot{\boldsymbol{s}}_2) + \{\boldsymbol{s}_2, \mathtt{N}_2(\boldsymbol{s}_2)\}, \\ \boldsymbol{w}_1 + \boldsymbol{R}_1 + \boldsymbol{G}_1 - \boldsymbol{w}_2 - \boldsymbol{R}_2 &= \mathtt{N}_1(\dot{\boldsymbol{s}}_1) + \{\boldsymbol{s}_1, \mathtt{N}_1(\boldsymbol{s}_1)\}. \end{aligned} \tag{11.97}$$

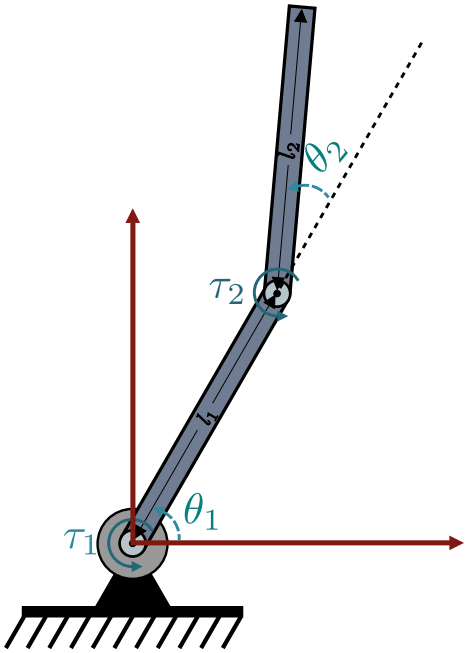


Fig. 11.3 A robot arm with two revoluted driven links

Reordering conveniently these equations for the doting with two screw lines $\boldsymbol{L}_2$ and $\boldsymbol{L}_1$

$$\begin{aligned} \boldsymbol{w}_2 + \boldsymbol{R}_2 &= \mathtt{N}_2(\dot{\boldsymbol{s}}_2) + \{\boldsymbol{s}_2, \mathtt{N}_2(\boldsymbol{s}_2)\} - \boldsymbol{G}_2, \\ \boldsymbol{w}_1 + \boldsymbol{R}_1 &= \mathtt{N}_1(\dot{\boldsymbol{s}}_1) + \{\boldsymbol{s}_1, \mathtt{N}_1(\boldsymbol{s}_1)\} + \mathtt{N}_2(\dot{\boldsymbol{s}}_2) + \{\boldsymbol{s}_2, \mathtt{N}_2(\boldsymbol{s}_2)\} - \boldsymbol{G}_1 - \boldsymbol{G}_2. \end{aligned} \tag{11.98}$$

Pairing with the joint screw lines $\boldsymbol{L}_2$ and $\boldsymbol{L}_1$ expressed as 6D vectors, one gets rid of the reaction wrenches leaving the magnitude of the torques as follows:

$$\begin{aligned} \tau_2 &= \mathtt{N}_2(\dot{\boldsymbol{s}}_2) \cdot \boldsymbol{L}_2 + \{\boldsymbol{s}_2, \mathtt{N}_2(\boldsymbol{s}_2)\} \cdot \boldsymbol{L}_2 - \boldsymbol{G}_2 \cdot \boldsymbol{L}_2, \\ \tau_1 &= \mathtt{N}_1(\dot{\boldsymbol{s}}_1) \cdot \boldsymbol{L}_1 + \{\boldsymbol{s}_1, \mathtt{N}_1(\boldsymbol{s}_1)\} \cdot \boldsymbol{L}_1 + \mathtt{N}_2(\dot{\boldsymbol{s}}_2) \cdot \boldsymbol{L}_1 + \\ &\quad + \{\boldsymbol{s}_2, \mathtt{N}_2(\boldsymbol{s}_2)\} \cdot \boldsymbol{L}_1 - (\boldsymbol{G}_1 + \boldsymbol{G}_2) \cdot \boldsymbol{L}_1. \end{aligned} \tag{11.99}$$

The velocity screw or twist $\boldsymbol{s}_i$ of the j-link is computed using the Jacobian J where its columns are the joint line screws $\boldsymbol{L}_i$, namely,

$$\boldsymbol{s}_j = J\dot{\theta} \equiv \left[\mathbf{L}_1\mathbf{L}_2 \ldots \mathbf{L}_j\right]\left[\dot{\theta}_1, \dot{\theta}_2, \ldots, \dot{\theta}_j\right]^T, \tag{11.100}$$

thus the velocities at the two links are

$$\boldsymbol{s}_1 = \dot{\theta}_1 \boldsymbol{L}_1, \tag{11.101}$$

$$\boldsymbol{s}_2 = \dot{\theta}_2 \boldsymbol{L}_2 + \dot{\theta}_1 \boldsymbol{L}_1. \tag{11.102}$$

The accelerations at each link are computed taking the time derivative of the velocities $\boldsymbol{s}_i$

$$\dot{\boldsymbol{s}}_1 = \ddot{\theta}_1 \boldsymbol{L}_1 + \dot{\theta}_1 \dot{\boldsymbol{L}}_1, \tag{11.103}$$
$$\dot{\boldsymbol{s}}_2 = \ddot{\theta}_1 \boldsymbol{L}_1 + \dot{\theta}_1 \dot{\boldsymbol{L}}_1 + \ddot{\theta}_2 \boldsymbol{L}_2 + \dot{\theta}_2 \dot{\boldsymbol{L}}_2. \tag{11.104}$$

According to Eq. (11.73), the derivative of a screw line using the matrix $\mathtt{M}_i$ for $SE(3)$ reads

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathtt{M}_i \begin{bmatrix} \boldsymbol{n}_j \\ \mathbf{m}_j \end{bmatrix} = [\mathbf{s}_i, \mathbf{L}_j], \tag{11.105}$$

so using motors, the derivative of a screw line is given by

$$\begin{aligned} \boldsymbol{L}_2(t) &= \boldsymbol{M}_1(t)\boldsymbol{L}_2(0)\widetilde{\boldsymbol{M}}_1(t), \\ \frac{d}{dt}\boldsymbol{L}_2(t) &= [\boldsymbol{s}_1, \boldsymbol{L}_2] = \dot{\theta}_1 [\boldsymbol{L}_1, \boldsymbol{L}_2]. \end{aligned} \tag{11.106}$$

The derivatives of the two line screws are then given by

$$\dot{\boldsymbol{L}}_1 = \frac{d}{dt}\boldsymbol{L}_1 = [\boldsymbol{s}_1, \boldsymbol{L}_1] = \dot{\theta}_1 [\boldsymbol{L}_1, \boldsymbol{L}_1] = 0, \tag{11.107}$$
$$\dot{\boldsymbol{L}}_2 = \frac{d}{dt}\boldsymbol{L}_2 = [\boldsymbol{s}_1, \boldsymbol{L}_2] = \dot{\theta}_1 [\boldsymbol{L}_1, \boldsymbol{L}_2]. \tag{11.108}$$

Thus, we can rewrite the accelerations at each link in terms of Lie brackets, namely,

$$\dot{\boldsymbol{s}}_1 = \ddot{\theta}_1 \boldsymbol{L}_1, \tag{11.109}$$
$$\dot{\boldsymbol{s}}_2 = \ddot{\theta}_1 \boldsymbol{L}_1 + \ddot{\theta}_2 \boldsymbol{L}_2 + \dot{\theta}_1 \dot{\theta}_2 [\boldsymbol{L}_1, \boldsymbol{L}_2]. \tag{11.110}$$

11.4.3 N Robot Links

We have shown the dynamics for one- and two-link robot links, the generalization for n-link robots is straightforward. Let us show the equations for N = 5, first we write the equation of motion for each link and then we can conveniently group them.

$$\begin{aligned} \boldsymbol{w}_5 + \boldsymbol{R}_5 + \boldsymbol{G}_5 &= \mathtt{N}_5(\dot{\boldsymbol{s}}_5) + \{\boldsymbol{s}_5, \mathtt{N}_5(\boldsymbol{s}_5)\}, \\ \boldsymbol{w}_4 + \boldsymbol{R}_4 + \boldsymbol{G}_4 - \boldsymbol{w}_5 - \boldsymbol{R}_5 &= \mathtt{N}_4(\dot{\boldsymbol{s}}_4) + \{\boldsymbol{s}_4, \mathtt{N}_4(\boldsymbol{s}_4\}, \\ \boldsymbol{w}_3 + \boldsymbol{R}_3 + \boldsymbol{G}_3 - \boldsymbol{w}_4 - \boldsymbol{R}_4 &= \mathtt{N}_3(\dot{\boldsymbol{s}}_3) + \{\boldsymbol{s}_3, \mathtt{N}_3(\boldsymbol{s}_3)\}, \\ \boldsymbol{w}_2 + \boldsymbol{R}_2 + \boldsymbol{G}_2 - \boldsymbol{w}_3 - \boldsymbol{R}_3 &= \mathtt{N}_2(\dot{\boldsymbol{s}}_2) + \{\boldsymbol{s}_2, \mathtt{N}_2(\boldsymbol{s}_2)\}, \\ \boldsymbol{w}_1 + \boldsymbol{R}_1 + \boldsymbol{G}_1 - \boldsymbol{w}_2 - \boldsymbol{R}_2 &= \mathtt{N}_1(\dot{\boldsymbol{s}}_1) + \{\boldsymbol{s}_1, \mathtt{N}_1(\boldsymbol{s}_1)\}. \end{aligned} \tag{11.111}$$

These equations can be rearranged as we did with the two-link robot, namely,

$$\boldsymbol{w}_5 + \boldsymbol{R}_5 = \mathrm{N}_5(\dot{\boldsymbol{s}}_5) + \{\boldsymbol{s}_5, \mathrm{N}_5(\boldsymbol{s}_5)\} - \boldsymbol{G}_5, \tag{11.112}$$

$$\boldsymbol{w}_4 + \boldsymbol{R}_4 = \mathrm{N}_4(\dot{\boldsymbol{s}}_4) + \mathrm{N}_5(\dot{\boldsymbol{s}}_5) + \{\boldsymbol{s}_4, \mathrm{N}_4(\boldsymbol{s}_4)\} + \{\boldsymbol{s}_5, \mathrm{N}_5(\boldsymbol{s}_5)\} - \boldsymbol{G}_5 - \boldsymbol{G}_4,$$

$$\vdots$$

$$\boldsymbol{w}_1 + \boldsymbol{R}_1 = \sum_{j=i}^{5} \left[\mathrm{N}_j(\dot{\boldsymbol{s}}_j) + \{\boldsymbol{s}_j, \mathrm{N}_j(\boldsymbol{s}_j)\} - \boldsymbol{G}_j\right], \tag{11.113}$$

where the velocity screw or twist $\boldsymbol{s}_j$ at the j-link is computed using Eq. (11.100)

$$\boldsymbol{s}_j = \sum_{k=1}^{j} \dot{\theta}_k \boldsymbol{L}_k. \tag{11.114}$$

Generalizing Eq. (11.110), the acceleration screw at the j-link is computed as follows:

$$\dot{\boldsymbol{s}}_j = \sum_{k=1}^{j} \left[\ddot{\theta}_k \boldsymbol{L}_k + \sum_{l=1}^{k-1} \dot{\theta}_l \dot{\theta}_k \left[\boldsymbol{L}_l, \boldsymbol{L}_k\right]\right]. \tag{11.115}$$

Recall that the momentum is computed as follows:

$$\mathcal{P}_j = N_j(\boldsymbol{s}_j). \tag{11.116}$$

According to Eq. (11.113), a i-wrench is computed as follows:

$$\boldsymbol{w}_i = \sum_{j=i}^{5} \left[\mathrm{N}_j(\dot{\boldsymbol{s}}_j) \cdot \boldsymbol{L}_i + \{\boldsymbol{s}_j, \mathrm{N}_j(\boldsymbol{s}_j)\} \cdot \boldsymbol{L}_i - \boldsymbol{G}_j \cdot \boldsymbol{L}_i\right] - \boldsymbol{R}_i. \tag{11.117}$$

Pairing each Eq. (11.113) with its respective screw line $\boldsymbol{L}_i$ to get rid of the reaction wrenches, we get the five joint torques as follows:

$$\tau_i = \sum_{j=i}^{5} \left[\mathrm{N}_j(\dot{\boldsymbol{s}}_j) \cdot \boldsymbol{L}_i + \{\boldsymbol{s}_j, \mathrm{N}_j(\boldsymbol{s}_j)\} \cdot \boldsymbol{L}_i - \boldsymbol{G}_j \cdot \boldsymbol{L}_i\right]. \tag{11.118}$$

Since the wrench due to the gravity on the j-link is a force acting on the link's center of mass, considering that the z-axis of the coordinate system is parallel to the gravity vector, the gravity wrench can be then written as

$$\boldsymbol{G}_j = -mg\boldsymbol{c}_i \wedge e_{\infty 3} - mge_{\infty 3} \equiv \begin{bmatrix} -mg\boldsymbol{c}_i \wedge \boldsymbol{e}_{\infty 3} \\ 0 \\ 0 \\ -mge_{\infty 3} \end{bmatrix}. \tag{11.119}$$

Considering the following screw

$$\boldsymbol{G} = -ge_{\infty 3} \equiv \mathbf{G} = \begin{bmatrix} \mathbf{0} \\ 0 \\ 0 \\ -g \end{bmatrix}, \tag{11.120}$$

one can show that the gravity screw $\boldsymbol{G}_j$ can be expressed in terms of the inertia matrix $\mathtt{N}$ as follows:

$$\boldsymbol{G}_j = \mathtt{N}_j(\boldsymbol{G}). \tag{11.121}$$

Finally, using this result, one can write the equations of motion even in a more compact manner as

$$\tau_i = \sum_{j=i}^{5} \left[\mathtt{N}_j(\dot{\boldsymbol{s}}_j - \boldsymbol{G}) \cdot \boldsymbol{L}_i + \{\boldsymbol{s}_j, \mathtt{N}_j \boldsymbol{s}_j\} \cdot \boldsymbol{L}_i \right]. \tag{11.122}$$

11.4.4 Recursive Algorithm

In previous subsection, we derived the general equations of motion for a 5-DOF robot manipulator with five links connected in series. These equations still are no usable to control the robot as how we will compute the required torques given: the joint angles, their first and second derivatives together with the home values of the inertia matrices and joint screws. The problem is that these equations are expressed in terms of the current values of the joint screws and inertia matrices. Thus, we would have to utilize the forward kinematics to compute the current .. in the equations. As a control algorithm seems to be very efficient for real-time robot control. To improve the algorithm, firstly, the calculations can be done in coordinates at rest relative to the links; in this way, the link inertias will be constant with respect to these coordinates. Secondly, the required coordinate transformation can be done recursively. Thus, the screw lines $\boldsymbol{L}_i$ and the screw velocities or twists $\dot{\boldsymbol{s}}_i$ are computed using the respective motors $\boldsymbol{M}_i$ as follows:

$$\boldsymbol{L}(\theta_i) = \prod_{k=1}^{i} \boldsymbol{M}_{i+1-k}(\theta_{i+1-k}) \boldsymbol{L}(0)_i \widetilde{\boldsymbol{M}}_{i+1-k}(\theta_{j+1-i}), \tag{11.123}$$

$$\boldsymbol{s}_i = \prod_{k=1}^{i} \boldsymbol{M}_{i+1-k}(\theta_{i+1-k}) \boldsymbol{s}_i^0 \widetilde{\boldsymbol{M}}_{i+1-k}(\theta_{j+1-i}), \tag{11.124}$$

where $\boldsymbol{s}_i^0$ is the velocity of the i-link expressed in coordinates at rest with respect to the link. The constant gravity screw $\mathbf{G}$ must also be transformed as

$$\boldsymbol{G} = \prod_{k=1}^{j} \boldsymbol{M}_{j+1-k}(\theta_{j+1-k}) \boldsymbol{G}_j^0 \widetilde{\boldsymbol{M}}_{j+1-k}(\theta_{j+1-i}). \tag{11.125}$$

In the case of the inertia, it is similarly computed in terms of matrices as follows:

$$\mathtt{N}_i = \prod_{k=1}^{i} (\mathtt{M}^T)_{i+1-k}^{-1} \mathtt{N}_i^0 (\mathtt{M})_{i+1-k}^{-1}, \tag{11.126}$$

where according to Eq. (11.65), the j-homogeneous matrix to transform a screw is given by

$$(\mathtt{M}^T)_j^{-1} = \begin{bmatrix} \mathtt{R}_j & [\mathbf{t}_j]_x \mathtt{R}_j \\ \mathtt{O} & \mathtt{R}_j \end{bmatrix}, \quad \mathtt{M}_j^{-1} = \begin{bmatrix} \mathtt{R}_j^T & \mathtt{O} \\ -\mathtt{R}_j^T [\mathbf{t}_j]_x & \mathtt{R}_j^T \end{bmatrix}, \tag{11.127}$$

so under the rotation and translation the first 6×6 inertia matrix $\mathtt{N}_1$ is transformed as

$$\mathtt{N}_1 = (\mathtt{M}_1^T)^{-1} \mathtt{N}_1^0 \mathtt{M}_1^{-1}. \tag{11.128}$$

The velocity screw or twist of the links is given in Eq. (11.100)

$$\boldsymbol{s}_j = \dot{\theta}_1 \boldsymbol{L}_1 + \dot{\theta}_2 \boldsymbol{L}_2 + \cdots + \dot{\theta}_l \boldsymbol{L}_j = \sum_{k=1}^{j} \dot{\theta}_k \boldsymbol{L}_k, \tag{11.129}$$

in terms of coordinates at rest relative to the j-link, the twist becomes

$$\boldsymbol{s}_j^0 = \dot{\theta}_1 \prod_{i=2}^{j} \widetilde{\boldsymbol{M}}_i \boldsymbol{L}_1^0 \boldsymbol{M}_i + \dot{\theta}_2 \prod_{i=3}^{j} \widetilde{\boldsymbol{M}}_i \boldsymbol{L}_2^0 \boldsymbol{M}_i + \cdots + \dot{\theta}_j \boldsymbol{L}_j^0. \tag{11.130}$$

We see the use of many motors $\boldsymbol{M}_i$ for the computing, and this equation can be simplified using a recursive procedure as follows:

$$\boldsymbol{s}_j^0 = \widetilde{\boldsymbol{M}}_j \boldsymbol{L}_{j-1}^0 \boldsymbol{M}_j + \dot{\theta}_j \boldsymbol{L}_j^0, \quad \boldsymbol{s}_1^0 = \dot{\theta}_1 \boldsymbol{L}_1^0. \tag{11.131}$$

The equation of the acceleration screw can be simplified in the same manner. Equation (11.115) of the acceleration screw of the j-link is

$$\dot{\boldsymbol{s}}_j = \ddot{\theta}_1 \boldsymbol{L}_1 + \ddot{\theta}_2 \boldsymbol{L}_2 + \cdots + \ddot{\theta}_j \boldsymbol{L}_j + \dot{\theta}_1 \dot{\theta}_2 [\boldsymbol{L}_1, \boldsymbol{L}_2] + \dot{\theta}_2 \dot{\theta}_3 [\boldsymbol{L}_2, \boldsymbol{L}_3] + \cdots + + \dot{\theta}_{j-1} \dot{\theta}_j \left[\boldsymbol{L}_{j-1}, \boldsymbol{L}_j\right]. \tag{11.132}$$

Using the coordinates at rest relative to the j-link, this becomes

$$\begin{aligned} \dot{\boldsymbol{s}}_j^0 = {} & \ddot{\theta}_1 \prod_{i=2}^{j} \widetilde{\boldsymbol{M}}_i \boldsymbol{L}_1^0 \boldsymbol{M}_i + \ddot{\theta}_2 \prod_{i=3}^{j} \widetilde{\boldsymbol{M}}_i \boldsymbol{L}_2^0 \boldsymbol{M}_i + \cdots + \ddot{\theta}_j \boldsymbol{L}_j^0 + \\ & + \dot{\theta}_1 \dot{\theta}_2 \prod_{i=3}^{j} \widetilde{\boldsymbol{M}}_i [\boldsymbol{L}_1, \boldsymbol{L}_2] \boldsymbol{M}_i + \cdots + \\ & + \dot{\theta}_{j-2} \dot{\theta}_{j-1} \widetilde{\boldsymbol{M}}_j \left[\boldsymbol{L}_{j-2}, \boldsymbol{L}_{j-1}\right] \boldsymbol{M}_j + \dot{\theta}_{j-1} \dot{\theta}_j \left[\boldsymbol{L}_{j-1}, \boldsymbol{L}_j\right]. \end{aligned} \tag{11.133}$$

Again, this can be simplified as a recursive procedure as follows:

$$\begin{aligned} \dot{\boldsymbol{s}}_j^0 &= \widetilde{\boldsymbol{M}}_j \dot{\boldsymbol{s}}_{j-1}^0 \boldsymbol{M}_j + \ddot{\theta}_j \boldsymbol{L}_j^0 + \dot{\theta}_{j-1} \dot{\theta}_j \left[\boldsymbol{L}_{j-1}, \boldsymbol{L}_j\right], \\ \dot{\boldsymbol{s}}_1^0 &= \ddot{\theta}_1 \boldsymbol{L}_1^0. \end{aligned} \tag{11.134}$$

Integrating in this equation the gravity screw, we get even a better equation

$$\begin{aligned} (\dot{\boldsymbol{s}}_j^0 - \boldsymbol{G}_j^0) &= \widetilde{\boldsymbol{M}}_j (\dot{\boldsymbol{s}}_{j-1}^0 - \boldsymbol{G}_{j-1}^0) \boldsymbol{M}_j + \dot{\theta}_j \boldsymbol{L}_j^0 + \dot{\theta}_{j-1} \dot{\theta}_j \left[\boldsymbol{L}_{j-1}, \boldsymbol{L}_j\right], \\ \dot{\boldsymbol{s}}_1^0 - \boldsymbol{G}_1^0 &= \ddot{\theta}_1 \boldsymbol{L}_1^0 - \boldsymbol{G}. \end{aligned} \tag{11.135}$$

Using these results, we can now formulate an efficient recursive algorithm which consists of the outward computing toward the robot end-effector and the inward computing toward the robot basis.

Outward computing:

• Compute the link velocity or twist $\boldsymbol{s}_j$ at each link

$$\begin{aligned} \boldsymbol{s}_1^0 &= \dot{\theta}_1 \boldsymbol{L}_1^0, \\ \boldsymbol{s}_j^0 &= \widetilde{\boldsymbol{M}}_j \boldsymbol{s}_{j-1}^0 \boldsymbol{M}_j + \dot{\theta}_j \boldsymbol{L}_j^0. \end{aligned} \tag{11.136}$$

• Compute the link acceleration $\dot{\boldsymbol{s}}_j$ at each link

$$\begin{aligned} \dot{\boldsymbol{s}}_1^0 - \boldsymbol{G}_1^0 &= \ddot{\theta}_1 \boldsymbol{L}_1^0 - \boldsymbol{G}, \\ (\dot{\boldsymbol{s}}_j^0 - \boldsymbol{G}_j^0) &= \widetilde{\boldsymbol{M}}_j (\dot{\boldsymbol{s}}_{j-1}^0 - \boldsymbol{G}_{j-1}^0) \boldsymbol{M}_j + \ddot{\theta}_j \boldsymbol{L}_j^0 + \dot{\theta}_j \left[\boldsymbol{s}_j^0, \boldsymbol{L}_j^0\right]. \end{aligned} \tag{11.137}$$

Inward computing:

• Compute the wrench $\boldsymbol{w}_j$ due to each link

$$\boldsymbol{w}_j^0 = N_j^0 (\dot{\boldsymbol{s}}_j^0 - \boldsymbol{G}_j^0) + \{\boldsymbol{s}_j^0, N_j^0 (\boldsymbol{s}_j^0)\}. \tag{11.138}$$

• Compute the total wrench acting $\boldsymbol{W}_i$ on each link

$$\begin{aligned} \boldsymbol{W}_K^0 &= \boldsymbol{w}_K^0, \\ \boldsymbol{W}_i^0 &= \widetilde{\boldsymbol{M}}_j \boldsymbol{W}_{j+1}^0 \boldsymbol{M}_j + \boldsymbol{w}_j^0. \end{aligned} \tag{11.139}$$

• Compute the torques at each link by pairing with the joint screws $\boldsymbol{L}_i^0$,

$$\tau_i = < \boldsymbol{W}_i^0 \boldsymbol{L}_i^0 >_0 = \boldsymbol{W}_i^0 \cdot \boldsymbol{L}_i^0. \tag{11.140}$$

As an illustration, we present an example of a 2D planar manipulator, $K = 2$. We wrote a program using the eClifford (R. Ablamowicz) Clifford algebra software package in Maple. We implement the above-formulated Newton–Euler iterative algorithm to compute the inertia matrix, centripetal-Coriolis vector, and gravity vector, respectively,

$$M = \begin{bmatrix} L_1 L_2 \cos(\theta_2) + \frac{1}{4} L_2^2 m_2 + \frac{1}{4} L_1^2 m_1 + L_1^2 m_2 & \frac{1}{4} L_2^2 m_2 + \frac{1}{2} L_1 L_2 m_2 \cos(\theta_2) \\ \frac{1}{4} L_2^2 m_2 + + \frac{1}{2} L_1 L_2 m_2 \cos(\theta_2) & \frac{1}{4} L_2^2 m_2 \end{bmatrix}$$

$$V = \begin{bmatrix} -L_1 L_2 \dot{\theta}_2 \sin(\theta_2)(\dot{\theta}_1 + \frac{1}{2}\dot{\theta}_2) + c\theta_1 + d\dot{\theta}_2 \\ \frac{1}{2} L_1 L_2 m_2 \sin(\theta_2) \end{bmatrix} \tag{11.141}$$

$$G = \begin{bmatrix} (\frac{1}{2} \sin(\theta 1 + \theta_2) L_2 m_2 + \frac{1}{2} L_1 m_1 \sin(\theta 1) + L_1 m_2 \sin(\theta_1) g \\ \frac{1}{2} \sin(\theta_1 + \theta_2) g L_2 m_2 \end{bmatrix}. \tag{11.142}$$

11.5 Euler–Lagrange Dynamics

In this section, we describe the equations of kinetic and potential energy in terms of geometric algebra. Based on these equations and using the Lagrange equation, we synthesize the dynamic model of any n-degrees-of-freedom serial robot.

11.5.1 Kinetic Energy

We introduce in our analysis the mass center in order to formulate an expression that describes the kinetic energy of a system of particles.

Kinetic energy of a system of particles. We are considering a system with n particles shown in Fig. 11.4. The total relative kinetic energy K of the system is given by

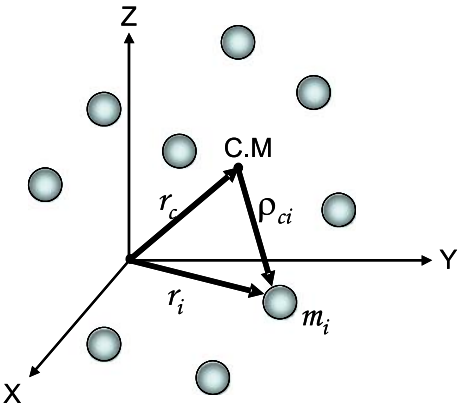


Fig. 11.4 System of particles with their mass center

$$K = \sum_{i=1}^{n} \frac{1}{2} m_i V_i^2. \tag{11.143}$$

Now we will rewrite Eq. (11.143) to introduce the mass center. Here r_i represents the distance to the particle, r_c the distance to the mass center, and ρ_{ci} the distance from the mass center to the particle:

$$r_i = r_c + \rho_{ci}. \tag{11.144}$$

The time derivative of Eq. (11.144) is

$$\dot{r}_i = \dot{r}_c + \dot{\rho}_{ci}. \tag{11.145}$$

Therefore, the velocity equation of the i-th particle ($\dot{\rho}_{ci}$) with respect to the mass center is given by

$$V_i = V_c + \dot{\rho}_{ci}. \tag{11.146}$$

By substitution of Eq. (11.146) in the expression of kinetic energy (11.143), we obtain

$$\begin{aligned} K &= \sum_{i=1}^{n} \frac{1}{2} m_i (V_c + \dot{\rho}_{ci})^2, \qquad (11.147) \\ &= \frac{1}{2} \sum_{i=1}^{n} m_i (V_c)^2 + \sum_{i=1}^{n} m_i V_c \dot{\rho}_{ci} + \frac{1}{2} \sum_{i=1}^{n} m_i (\dot{\rho}_{ci})^2. \end{aligned}$$

As V_c is not related to the sum index i, we can extract it as given below:

$$K = \frac{1}{2}V_c^2\left(\sum_{i=1}^{n} m_i\right) + V_c\frac{d}{dt}\sum_{i=1}^{n} m_i\rho_{ci} + \frac{1}{2}\sum_{i=1}^{n} m_i\dot{\rho}_{ci}^2. \tag{11.148}$$

As $m = \sum_{i=1}^{n} m_i$ is the total mass of the system and considering that $\sum_{i=1}^{n} m_i\rho_{ci}$ is by definition equal to zero,

$$K = \frac{1}{2}mV_c^2 + \frac{1}{2}\sum_{i=1}^{n} m_i\dot{\rho}_{ci}^2. \tag{11.149}$$

As conclusion, we see that *kinetic energy* with respect to a reference system could be considered as the sum of two parts: *(1) the kinetic energy of total mass moves with respect to this reference system at the same velocity* plus *(2) the kinetic energy of the particles moves with respect to the mass center (momentum of inertia).*

The kinetic energy of a robot arm. We denote by x_i the mass center i in its initial position and by $\boldsymbol{x}_i'$ the mass center as function of joint variables. Similarly, we will denote the joint axis i as $\boldsymbol{L}_i$ and the joint axis i as a function of the joint variables as $\boldsymbol{L}_i'$.

Recall the direct kinematics Eq. (6.131) that relates x_i to $\boldsymbol{x}_i'$ and L to $\boldsymbol{L}_i'$ and is written using conformal geometric algebras as

$$\boldsymbol{x}_i' = \boldsymbol{M}_1\boldsymbol{M}_2\cdots\boldsymbol{M}_i x_i\widetilde{\boldsymbol{M}}_i\cdots\widetilde{\boldsymbol{M}}_2\widetilde{\boldsymbol{M}}_1, \tag{11.150}$$

$$\boldsymbol{L}_i' = \boldsymbol{M}_1\boldsymbol{M}_2\cdots\boldsymbol{M}_{i-1}\boldsymbol{L}_i\widetilde{\boldsymbol{M}}_{i-1}\cdots\widetilde{\boldsymbol{M}}_2\widetilde{\boldsymbol{M}}_1. \tag{11.151}$$

We have seen that the kinetic energy is equal to the sum of the energy related to the velocity of mass center and the energy related to the momentum of inertia. So the kinetic energy of the link i is computed as

$$K_i = \frac{1}{2}m_i\dot{\boldsymbol{x}}_i'^2 + \frac{1}{2}I_i\left(\sum_{j=1}^{i}\dot{q}_j\right)^2, \tag{11.152}$$

where I_i is the inertia of the link i and $\dot{\boldsymbol{x}}_i'$ represents the velocity of the mass center $\boldsymbol{x}_i'$. The velocity of the mass center is computed using the equation of differential kinematics (5.98) explained in Sect. 5.9 (Fig. 11.5):

$$\dot{\boldsymbol{x}}'_i = \boldsymbol{x}_i'\cdot\left(\sum_{j=1}^{i}\boldsymbol{L}_j'\dot{q}_j\right). \tag{11.153}$$

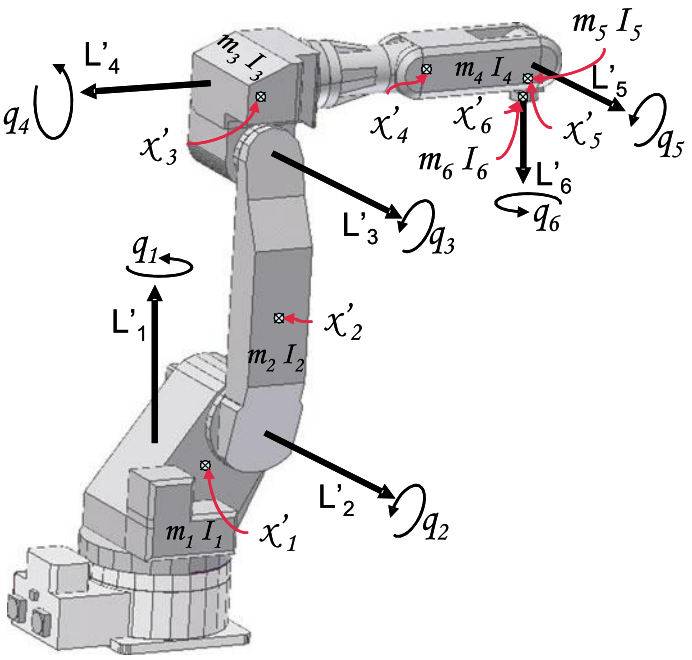


Fig. 11.5 Robot arm AdepdtSix600

Replacing Eq. (11.153) in (11.152), we have the expression of kinetic energy in conformal geometric algebra:

$$K_i = \frac{1}{2} m_i \left[\boldsymbol{x}_i' \cdot \left(\sum_{j=1}^{i} \boldsymbol{L}_j' \dot{q}_j \right) \right]^2 + \frac{1}{2} I_i \left(\sum_{j=1}^{i} \dot{q}_j \right)^2 . \tag{11.154}$$

The total kinetic energy on the arm is given by the expression $\sum_{i=1}^{n} K_i$, where n is the number of degrees of freedom. In order to simplify the explanation, we will separate the kinetic energy $K = K_v + K_I$ into two components, K_v and K_I, defined as

$$K_v = \frac{1}{2} \sum_{i=1}^{n} m_i \left[\boldsymbol{x}_i' \cdot \left(\sum_{j=1}^{i} \boldsymbol{L}_j' \dot{q}_j \right) \right]^2 , \tag{11.155}$$

$$K_I = \frac{1}{2} \sum_{i=1}^{n} I_i \left(\sum_{j=1}^{i} \dot{q}_j \right)^2 . \tag{11.156}$$

We will attend first to K_v and later to K_I. The objective is to simplify the expression of total kinetic energy in the arm:

$$K_v = \frac{1}{2}\sum_{i=1}^{n} m_i \left[\boldsymbol{x}_i' \cdot \left(\sum_{j=1}^{i} \boldsymbol{L}_j' \dot{q}_j \right) \right]^2 . \tag{11.157}$$

The square of the velocity's magnitude is equal to the dot product of the vector with itself:

$$K_v = \frac{1}{2}\sum_{i=1}^{n} m_i \left[\boldsymbol{x}_i' \cdot \left(\sum_{j=1}^{i} \boldsymbol{L}_j' \dot{q}_j \right) \right] \cdot \left[\boldsymbol{x}_i' \cdot \left(\sum_{j=1}^{i} \boldsymbol{L}_j' \dot{q}_j \right) \right], \tag{11.158}$$

introducing the term $\boldsymbol{x}_i'$ into the sum,

$$K_v = \frac{1}{2}\sum_{i=1}^{n} m_i \left(\sum_{j=1}^{i} (\boldsymbol{x}_i' \cdot \boldsymbol{L}_j') \dot{q}_j \right) \cdot \left(\sum_{j=1}^{i} (\boldsymbol{x}_i' \cdot \boldsymbol{L}_j') \dot{q}_j \right). \tag{11.159}$$

Evaluating the sum for j from 1 to i gives

$$\begin{aligned} K_v = \frac{1}{2}\sum_{i=1}^{n} m_i & \left(\boldsymbol{x}_i' \cdot \boldsymbol{L}_1' \dot{q}_1 + \cdots + \boldsymbol{x}_i' \cdot \boldsymbol{L}_i' \dot{q}_i \right) \\ & \cdot \left(\boldsymbol{x}_i' \cdot \boldsymbol{L}_1' \dot{q}_1 + \cdots + \boldsymbol{x}_i' \cdot \boldsymbol{L}_i' \dot{q}_i \right). \end{aligned} \tag{11.160}$$

Evaluating the sum for i from 1 to n gives

$$\begin{aligned} K_v = \frac{1}{2} [\ & m_1 (\boldsymbol{x}_1' \cdot \boldsymbol{L}_1' \dot{q}_1) \cdot (\boldsymbol{x}_1' \cdot \boldsymbol{L}_1' \dot{q}_1) + \cdots \\ & m_2 (\boldsymbol{x}_2' \cdot \boldsymbol{L}_1' \dot{q}_1 + \boldsymbol{x}_2' \cdot \boldsymbol{L}_2' \dot{q}_2) \cdot (\boldsymbol{x}_2' \cdot \boldsymbol{L}_1' \dot{q}_1 + \boldsymbol{x}_2' \cdot \boldsymbol{L}_2' \dot{q}_2) + \cdots \\ & \vdots \\ & m_n \left(\boldsymbol{x}_n' \cdot \boldsymbol{L}_1' \dot{q}_1 + \cdots + \boldsymbol{x}_n' \cdot \boldsymbol{L}_n' \dot{q}_n \right) \cdot \left(\boldsymbol{x}_n' \cdot \boldsymbol{L}_1' \dot{q}_1 + \cdots + \boldsymbol{x}_n' \cdot \boldsymbol{L}_n' \dot{q}_n \right)]. \end{aligned} \tag{11.161}$$

Reorganizing the terms and extracting $\dot{q}$, we have

$$\begin{aligned} 2K_v = & \left(m_1 (\boldsymbol{x}_1' \cdot \boldsymbol{L}_1') \cdot (\boldsymbol{x}_1' \cdot \boldsymbol{L}_1') + \cdots + m_n (\boldsymbol{x}_n' \cdot \boldsymbol{L}_1') \cdot (\boldsymbol{x}_n' \cdot \boldsymbol{L}_1') \right) \dot{q}_1^2 + \\ & \left(m_1 (\boldsymbol{x}_1' \cdot \boldsymbol{L}_2') \cdot (\boldsymbol{x}_1' \cdot \boldsymbol{L}_2') + \cdots + m_n (\boldsymbol{x}_n' \cdot \boldsymbol{L}_2') \cdot (\boldsymbol{x}_n' \cdot \boldsymbol{L}_2') \right) \dot{q}_2^2 + \\ & \qquad \vdots \qquad\qquad\qquad \vdots \qquad\qquad\qquad \vdots \\ & \left(m_1 (\boldsymbol{x}_1' \cdot \boldsymbol{L}_n') \cdot (\boldsymbol{x}_1' \cdot \boldsymbol{L}_n') + \cdots + m_n (\boldsymbol{x}_n' \cdot \boldsymbol{L}_n') \cdot (\boldsymbol{x}_n' \cdot \boldsymbol{L}_n') \right) \dot{q}_n^2 + \\ & \left(m_2 (\boldsymbol{x}_2' \cdot \boldsymbol{L}_1') \cdot (\boldsymbol{x}_2' \cdot \boldsymbol{L}_2') + \cdots + m_2 (\boldsymbol{x}_n' \cdot \boldsymbol{L}_1') \cdot (\boldsymbol{x}_n' \cdot \boldsymbol{L}_2') \right) \dot{q}_1 \dot{q}_2 + \end{aligned}$$

$$\left(m_3(\boldsymbol{x}_3' \cdot \boldsymbol{L}_1') \cdot (\boldsymbol{x}_3' \cdot \boldsymbol{L}_3') + \cdots + m_2(\boldsymbol{x}_n' \cdot \boldsymbol{L}_1') \cdot (\boldsymbol{x}_n' \cdot \boldsymbol{L}_3')\right) \dot{q}_1 \dot{q}_3 + \\ \vdots \\ m_n(\boldsymbol{x}_n' \cdot \boldsymbol{L}_{n-1}') \cdot (\boldsymbol{x}_n' \cdot \boldsymbol{L}_n') \dot{q}_{n-1} \dot{q}_n. \quad (11.162)$$

Thanks to this decomposition, it is easy to see the symmetry of the terms: they could be organized in a matrix form to get a better compression of this equation:

$$K_v = \frac{1}{2} \begin{pmatrix} \dot{q}_1 & \cdots & \dot{q}_n \end{pmatrix} \mathcal{M}_v \begin{pmatrix} \dot{q}_1 \\ \vdots \\ \dot{q}_n \end{pmatrix}, \quad (11.163)$$

where $\mathcal{M}_v$ is equal to

$$\mathcal{M}_v = \frac{1}{2} \begin{pmatrix} \sum_{j=1}^{n} m_j(\boldsymbol{x}_j' \cdot \boldsymbol{L}_1') \cdot (\boldsymbol{x}_j' \cdot \boldsymbol{L}_1') & \cdots & \sum_{j=n}^{n} m_j(\boldsymbol{x}_j' \cdot \boldsymbol{L}_1') \cdot (\boldsymbol{x}_j' \cdot \boldsymbol{L}_n') \\ \sum_{j=2}^{n} m_j(\boldsymbol{x}_j' \cdot \boldsymbol{L}_2') \cdot (\boldsymbol{x}_j' \cdot \boldsymbol{L}_1') & \cdots & \sum_{j=n}^{n} m_j(\boldsymbol{x}_j' \cdot \boldsymbol{L}_2') \cdot (\boldsymbol{x}_j' \cdot \boldsymbol{L}_n') \\ \vdots & \ddots & \vdots \\ \sum_{j=n}^{n} m_j(\boldsymbol{x}_j' \cdot \boldsymbol{L}_n') \cdot (\boldsymbol{x}_j' \cdot \boldsymbol{L}_1') & \cdots & \sum_{j=n}^{n} m_j(\boldsymbol{x}_j' \cdot \boldsymbol{L}_n') \cdot (\boldsymbol{x}_j' \cdot \boldsymbol{L}_n') \end{pmatrix}, \quad (11.164)$$

and each element $\mathcal{M}_{v_{ij}}$ of the matrix $\mathcal{M}_v$ is computed using

$$\mathcal{M}_{v_{ij}} = \sum_{k=Max(i,j)}^{n} m_k(\boldsymbol{x}_k' \cdot \boldsymbol{L}_i') \cdot (\boldsymbol{x}_k' \cdot \boldsymbol{L}_j'). \quad (11.165)$$

Note that the symmetric matrix $\mathcal{M}_v$ could be separated in the product of three matrices, two triangular and one diagonal:

$$\mathcal{M}_v = \begin{pmatrix} \boldsymbol{x}_1' \cdot \boldsymbol{L}_1' & \boldsymbol{x}_2' \cdot \boldsymbol{L}_1' & \cdots & \boldsymbol{x}_n' \cdot \boldsymbol{L}_1' \\ 0 & \boldsymbol{x}_2' \cdot \boldsymbol{L}_2' & \cdots & \boldsymbol{x}_n' \cdot \boldsymbol{L}_2' \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \boldsymbol{x}_n' \cdot \boldsymbol{L}_n' \end{pmatrix} \begin{pmatrix} m_1 & 0 & \cdots & 0 \\ 0 & m_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & m_n \end{pmatrix}$$

$$\begin{pmatrix} \boldsymbol{x}_1' \cdot \boldsymbol{L}_1' & 0 & \cdots & 0 \\ \boldsymbol{x}_2' \cdot \boldsymbol{L}_1' & \boldsymbol{x}_2' \cdot \boldsymbol{L}_2' & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{x}_n' \cdot \boldsymbol{L}_1' & \boldsymbol{x}_n' \cdot \boldsymbol{L}_2' & \cdots & \boldsymbol{x}_n' \cdot \boldsymbol{L}_n' \end{pmatrix}. \quad (11.166)$$

Now we will define the matrix m and the matrix V as

$$m := \begin{pmatrix} m_1 & 0 & \cdots & 0 \\ 0 & m_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & m_n \end{pmatrix}, \tag{11.167}$$

$$V := \begin{pmatrix} \boldsymbol{x}_1' \cdot \boldsymbol{L}_1' & 0 & \cdots & 0 \\ \boldsymbol{x}_2' \cdot \boldsymbol{L}_1' & \boldsymbol{x}_2' \cdot \boldsymbol{L}_2' & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{x}_n' \cdot \boldsymbol{L}_1' & \boldsymbol{x}_n' \cdot \boldsymbol{L}_2' & \cdots & \boldsymbol{x}_n' \cdot \boldsymbol{L}_n' \end{pmatrix}. \tag{11.168}$$

Then we can write Eq. (11.166) as follows:

$$\mathcal{M}_v = V^T m V. \tag{11.169}$$

The elements of the matrix V are vectors and the elements of m are scalars, which means that the contribution of kinetic energy produced due to mass displacements with respect to the reference frame could be easily computed as

$$K_v = \frac{1}{2} \dot{q}^T V^T m V \dot{q}. \tag{11.170}$$

Now we follow a similar procedure for the component of the kinetic energy K_i:

$$K_I = \frac{1}{2} \sum_{i=1}^{n} I_i \left(\sum_{j=1}^{i} \dot{q}_j \right)^2. \tag{11.171}$$

Evaluating the sum for i and j from 1 to n, we get

$$K_I = \frac{1}{2} \left[I_1 (\dot{q}_1)^2 + I_2 (\dot{q}_1 + \dot{q}_2)^2 + \cdots + I_n (\dot{q}_1 + \cdots + \dot{q}_n)^2 \right]. \tag{11.172}$$

Expanding the expression, extracting $\dot{q}$, and writing in matrix form give us

$$K_I = \frac{1}{2} \begin{pmatrix} \dot{q}_1 & \cdots & \dot{q}_n \end{pmatrix} \mathcal{M}_I \begin{pmatrix} \dot{q}_1 \\ \vdots \\ \dot{q}_n \end{pmatrix}, \tag{11.173}$$

where

$$\mathcal{M}_I = \begin{pmatrix} \sum_{i=1}^{n} I_i & \sum_{i=2}^{n} I_i & \cdots & \sum_{i=n}^{n} I_i \\ \sum_{i=2}^{n} I_i & \sum_{i=2}^{n} I_i & \cdots & \sum_{i=n}^{n} I_i \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{i=n}^{n} I_i & \sum_{i=n}^{n} I_i & \cdots & \sum_{i=n}^{n} I_i \end{pmatrix}. \tag{11.174}$$

The matrix $\mathcal{M}_I$ can be written as the product of two matrices δ and I if we define them as

$$M_I = \delta I = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ 0 & 1 & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix} \begin{pmatrix} I_1 & 0 & \cdots & 0 \\ I_2 & I_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ I_n & I_n & \cdots & I_n \end{pmatrix}. \tag{11.175}$$

In such a way, the component of kinetic energy due to the movement of links around their mass center is given by

$$K_I = \frac{1}{2}\dot{q}^T \delta I \dot{q}. \tag{11.176}$$

In conclusion, we have an expression to compute the total kinetic energy of the serial robot using the axes of the robot and the mass centers in conformal geometric algebra:

$$K = \frac{1}{2}\dot{q}^T (V^T m V + \delta I)\dot{q}. \tag{11.177}$$

Note that this expression allows us to compute the kinetic energy without the derivatives.

11.5.2 Potential Energy

In contrast to kinetic energy, the potential energy does not depend on the velocity, but it depends on the position of each link of the serial robot. Thanks to the equations of direct kinematics (6.131) we can compute the position $\boldsymbol{x}_i'$ of each link. In order to know the potential energy U_i, we compute the dot product of these points and the force applied to each point:

$$U_i = \boldsymbol{x}_i' \cdot F_i. \tag{11.178}$$

Here the potential energy is due to conservative forces such as the gravity forces, then $F_i = m_i g e_2$. Also, the total potential energy of the system is equal to the sum of all U_i:

$$U = \sum_{i=1}^{n} \boldsymbol{x}_i' \cdot F_i. \tag{11.179}$$

11.5.3 Lagrange's Equations

The dynamic equations of a robot could be computed based on Newton's equations, but the formulation becomes complicated when the number of degrees of freedom increases. For this reason, we will use Lagrange's equations of movement.

The Lagrangian £ is defined as the difference between the kinetic energy and potential energy of the system:

$$£ = K - U. \tag{11.180}$$

Lagrange's equations of movement are given by

$$\frac{d}{dt}\left[\frac{\partial £}{\partial \dot{q}}\right] - \frac{\partial £}{\partial q} = \tau. \tag{11.181}$$

We first compute the partial derivative of £ with respect to $\dot{q}$:

$$\frac{\partial £}{\partial \dot{q}} = \frac{\partial K}{\partial \dot{q}} - \frac{\partial U}{\partial \dot{q}} = \frac{\partial K}{\partial \dot{q}}. \tag{11.182}$$

Note that the partial derivative of U with respect to $\dot{q}$ is always zero since U does not depend on the velocities of the joints $\dot{q}$. Replacing K in Eqs. (11.182) and (11.177) gives

$$\frac{\partial £}{\partial \dot{q}} = \frac{\partial}{\partial \dot{q}}\left(\frac{1}{2}\dot{q}^T(V^T m V + \delta I)\dot{q}\right) = (V^T m V + \delta I)\dot{q}. \tag{11.183}$$

In order to simplify the notation, the matrix $\mathcal{M}$ is defined as $\mathcal{M} = \mathcal{M}_v + \mathcal{M}_I = V^T m V + \delta I$ and Eq. (11.183) is now written as

$$\frac{\partial £}{\partial \dot{q}} = \mathcal{M}\dot{q}. \tag{11.184}$$

On the other hand, the partial derivative of £ with respect to q is given by

$$\frac{\partial £}{\partial q} = \frac{\partial K}{\partial q} - \frac{\partial U}{\partial q} = \frac{1}{2}\dot{q}^T\left(\frac{\partial \mathcal{M}}{\partial q}\right)\dot{q} - \frac{\partial U}{\partial q}. \tag{11.185}$$

Replacing Eqs. (11.184) and (11.185) in the Lagrange equation (11.181), we get

$$\frac{d}{dt}\left[\mathcal{M}\dot{q}\right] - \left[\frac{1}{2}\dot{q}^T\left(\frac{\partial \mathcal{M}}{\partial q}\right)\dot{q} - \frac{\partial U}{\partial q}\right] = \tau. \tag{11.186}$$

The time derivative of Eq. (11.186) gives

$$\mathcal{M}\ddot{q} + \dot{\mathcal{M}}\dot{q} - \frac{1}{2}\dot{q}^T\left(\frac{\partial \mathcal{M}}{\partial q}\right)\dot{q} + \frac{\partial U}{\partial q} = \tau. \tag{11.187}$$

In order to simplify the expression, we rename parts of the equation as follows:

$$C = \dot{\mathcal{M}} - \frac{1}{2}\dot{q}^T\left(\frac{\partial \mathcal{M}}{\partial q}\right), \tag{11.188}$$

$$G = \frac{\partial U}{\partial q}, \tag{11.189}$$

where C is the Coriolis and centrifugal matrix and G is the vector of gravitational components. Therefore, we can write the dynamic equation for a serial robot with n degrees of freedom:

$$\mathcal{M}\ddot{q} + C\dot{q} + G = \tau. \tag{11.190}$$

Now we analyze the G matrix, looking for an equation that allows us to get it without partial derivatives. Using Eq. (11.179), we write

$$G = \frac{\partial U}{\partial q} = \frac{\partial}{\partial q}\left(\sum_{i=1}^{n} F_i \cdot \boldsymbol{x}_i'\right). \tag{11.191}$$

Since the forces $F_i = m_i g e_2$ are produced by gravity, they do not depend on the joints positions q:

$$G = \sum_{i=1}^{n} F_i \cdot \left(\frac{\partial}{\partial q}\boldsymbol{x}_i'\right). \tag{11.192}$$

Recalling the equation of differential kinematics (5.98), we know that

$$\frac{\partial}{\partial q}\boldsymbol{x}_i' = \begin{pmatrix} \boldsymbol{x}_i' \cdot \boldsymbol{L}_1' \\ \boldsymbol{x}_i' \cdot \boldsymbol{L}_2' \\ \vdots \\ \boldsymbol{x}_i' \cdot \boldsymbol{L}_i' \end{pmatrix}. \tag{11.193}$$

Evaluating the sum of Eq. (11.192) from $i = 1$ to n and introducing on each term the evaluation of Eq. (11.193) give us

$$G = \begin{pmatrix} \boldsymbol{x}_1' \cdot \boldsymbol{L}_1' \\ 0 \\ \vdots \\ 0 \end{pmatrix} F_1 + \begin{pmatrix} \boldsymbol{x}_2' \cdot \boldsymbol{L}_1' \\ \boldsymbol{x}_2' \cdot \boldsymbol{L}_2' \\ \vdots \\ 0 \end{pmatrix} F_2 + \cdots + \begin{pmatrix} \boldsymbol{x}_n' \cdot \boldsymbol{L}_1' \\ \boldsymbol{x}_n' \cdot \boldsymbol{L}_2' \\ \vdots \\ \boldsymbol{x}_n' \cdot \boldsymbol{L}_n' \end{pmatrix} F_n. \tag{11.194}$$

Equation (11.194) is written as a matrix:

$$G = \begin{pmatrix} \boldsymbol{x}_1' \cdot \boldsymbol{L}_1' & \boldsymbol{x}_2' \cdot \boldsymbol{L}_1' & \cdots & \boldsymbol{x}_n' \cdot \boldsymbol{L}_1' \\ 0 & \boldsymbol{x}_2' \cdot \boldsymbol{L}_2' & \cdots & \boldsymbol{x}_n' \cdot \boldsymbol{L}_2' \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \boldsymbol{x}_n' \cdot \boldsymbol{L}_n' \end{pmatrix} \begin{pmatrix} F_1 \\ F_2 \\ \vdots \\ F_n \end{pmatrix}. \tag{11.195}$$

As one can see, this matrix is basically the transpose of the matrix V given in Eq. (11.168). Calling F the vector with components F_i, we can finally write the equation as given below:

$$G = V^T F. \tag{11.196}$$

Furthermore, F is given by the product of two matrices

$$F = \begin{pmatrix} m_1 & 0 & \cdots & 0 \\ 0 & m_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & m_n \end{pmatrix} \begin{pmatrix} ge_2 \\ ge_2 \\ \vdots \\ ge_2 \end{pmatrix} = ma, \tag{11.197}$$

where a is a vector of accelerations. Equation (11.197) allows us to separate the constant matrices and the variables of the serial robot:

$$G = V^T ma. \tag{11.198}$$

Finally, we have a short and useful equation to compute the vector G using the information of the joint's axes. Now we analyze the Coriolis matrix C. In fact, there are many ways to compute this matrix and there are many matrices C that satisfy the dynamic Eq. (11.190). Although we already have an equation to compute the matrix C (11.188), we will look for a simpler equation to avoid the necessity of derivatives. Based on the properties of the matrices $\mathcal{M}$ and C, we know that

$$\dot{\mathcal{M}} = C + C^T. \tag{11.199}$$

On the other hand, we know

$$\mathcal{M} = V^T m V + \delta I. \tag{11.200}$$

Computing the time derivative of $\mathcal{M}$ produces

$$\dot{\mathcal{M}} = \frac{d}{dt} M = \frac{d}{dt}\left(V^T m V + \delta I\right), \tag{11.201}$$

$$\dot{\mathcal{M}} = \frac{d}{dt} V^T m V, \tag{11.202}$$

$$\dot{\mathcal{M}} = V^T m \dot{V} + \dot{V}^T m V, \tag{11.203}$$

$$\dot{\mathcal{M}} = V^T m \dot{V} + (V^T m \dot{V})^T. \tag{11.204}$$

Taking into account Eqs. (11.199) and (11.204), we have a short and clear equation to compute the matrix C without derivatives:

$$C = V^T m \dot{V}. \tag{11.205}$$

The last sentence is true since we can compute the matrix $\dot{V}$ without derivatives just as a function of the joint's values $q, \dot{q}$ and the axes of the robot. In order to compute the element $\dot{V}_{ij}$, the time derivative of $\boldsymbol{x}'_i \cdot \boldsymbol{L}'_j$ is needed, using Eq. (11.153):

$$\dot{V}_{ij} = \frac{d}{dt}(\boldsymbol{x}'_i \boldsymbol{L}'_j) = \dot{\boldsymbol{x}}'_i \cdot \boldsymbol{L}'_j + \boldsymbol{x}'_i \cdot \dot{L}'_j, \tag{11.206}$$

$$\dot{V}_{ij} = \sum_{k=1}^{i} (\boldsymbol{x}'_i \cdot \boldsymbol{L}'_k) \cdot \boldsymbol{L}'_j \dot{q}_k + \frac{1}{2} \sum_{k=1}^{j-1} \boldsymbol{x}'_i \cdot (\boldsymbol{L}'_j \boldsymbol{L}'_k - \boldsymbol{L}'_k \boldsymbol{L}'_j) \dot{q}_k. \tag{11.207}$$

Note that $V_{ij} = 0$ whenever $j > i$ because $\dot{V}_{ij} = 0$. Perhaps these equations to get $\dot{V}$ could be confused. We will rewrite these equations as matrices to give clearer explanation of the method to compute $\dot{V}$. It is possible to write the matrix V as the product of two matrices:

$$V = \begin{pmatrix} \boldsymbol{x}'_1 & 0 & \cdots & 0 \\ 0 & \boldsymbol{x}'_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \boldsymbol{x}'_n \end{pmatrix} \begin{pmatrix} \boldsymbol{L}'_1 & 0 & \cdots & 0 \\ \boldsymbol{L}'_1 & \boldsymbol{L}'_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{L}'_1 & \boldsymbol{L}'_2 & \cdots & \boldsymbol{L}'_n \end{pmatrix} = XL, \tag{11.208}$$

then

$$\dot{V} = \dot{X} L + X \dot{L}, \tag{11.209}$$

with

$$\dot{X} = \begin{pmatrix} \dot{\boldsymbol{x}}_1' & 0 & \cdots & 0 \\ 0 & \dot{\boldsymbol{x}}_2' & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \dot{\boldsymbol{x}}_n' \end{pmatrix}. \tag{11.210}$$

Computing $\dot{\boldsymbol{x}}_i'$ is simple using the equation of differential kinematics (5.98):

$$\begin{pmatrix} \dot{\boldsymbol{x}}_1' \\ \vdots \\ \dot{\boldsymbol{x}}_n' \end{pmatrix} = XL\dot{q} = V\dot{q}. \tag{11.211}$$

On the other hand,

$$\dot{L}' = \begin{pmatrix} \dot{L}_1' & 0 & \cdots & 0 \\ \dot{L}_1' & \dot{L}_2' & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \dot{L}_1' & \dot{L}_2' & \cdots & \dot{L}_n' \end{pmatrix}. \tag{11.212}$$

To compute $\dot{L}_i'$, which represents the velocity of the axis i produced by the rotation around the previous axes, we can do the following:

$$\begin{pmatrix} \dot{L}_1' \\ \dot{L}_1' \\ \vdots \\ \dot{L}_n' \end{pmatrix} = \frac{1}{2} \left[\begin{pmatrix} \boldsymbol{L}_1'\boldsymbol{L}_1' & 0 & \cdots & 0 \\ \boldsymbol{L}_2'\boldsymbol{L}_1' & \boldsymbol{L}_2'\boldsymbol{L}_2' & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{L}_n'\boldsymbol{L}_1' & \boldsymbol{L}_n'\boldsymbol{L}_2' & \cdots & \boldsymbol{L}_n'\boldsymbol{L}_n' \end{pmatrix} \right.$$

$$\left. - \begin{pmatrix} \boldsymbol{L}_1'\boldsymbol{L}_1' & 0 & \cdots & 0 \\ \boldsymbol{L}_1'\boldsymbol{L}_2' & \boldsymbol{L}_2'\boldsymbol{L}_2' & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{L}_1'\boldsymbol{L}_n' & \boldsymbol{L}_2'\boldsymbol{L}_n' & \cdots & \boldsymbol{L}_n'\boldsymbol{L}_n' \end{pmatrix} \right] \dot{q}, \tag{11.213}$$

$$\begin{pmatrix} \dot{L}_1' \\ \dot{L}_1' \\ \vdots \\ \dot{L}_n' \end{pmatrix} = \frac{1}{2} \left[\begin{pmatrix} 0 & 0 & \cdots 0 \\ \boldsymbol{L}_2' \underline{\times} \boldsymbol{L}_1' & 0 & \cdots 0 \\ \vdots & \vdots & \ddots \vdots \\ \boldsymbol{L}_n' \underline{\times} \boldsymbol{L}_1' & \boldsymbol{L}_n' \underline{\times} \boldsymbol{L}_2' & \cdots 0 \end{pmatrix} \right] \dot{q}. \tag{11.214}$$

In conclusion, using Eqs. (11.183) and (11.205), we have rewritten the dynamic equation of serial robot with n degrees of freedom:

$$(V^T m V + \delta I)\ddot{q} + V^T m \dot{V}\dot{q} + V^T F = \tau, \tag{11.215}$$

$$\delta I\ddot{q} + V^T(mV\ddot{q} + m\dot{V}\dot{q} + F) = \tau. \tag{11.216}$$

This decomposition allows us to see the components of inertia momentum, centrifuge and gravity forces. Replacing F with ma and extracting m give

$$\delta I\ddot{q} + V^T m(V\ddot{q} + \dot{V}\dot{q} + a) = \tau. \tag{11.217}$$

Finally, Eq. (11.217) is the dynamic equation of n-degrees-of-freedom serial robot where the elements of the matrices are multivectors of the geometric algebra $G_{4,1,0}$. δ and a are constant and known matrices while m and I depend on the robot parameters but are also constants. Only the matrix V (Eq. (11.208)) and therefore $\dot{V}$ (Eq. 11.209)) changes over time.

Example 1 As an example, we will compare the classical and the proposed approach. We will compute the dynamics of a two-degree-of-freedom robot arm, see Fig. 11.6. First, we compute the matrices V and $\dot{V}$. These matrices will be used to compute $\mathcal{M}$, C, and G:

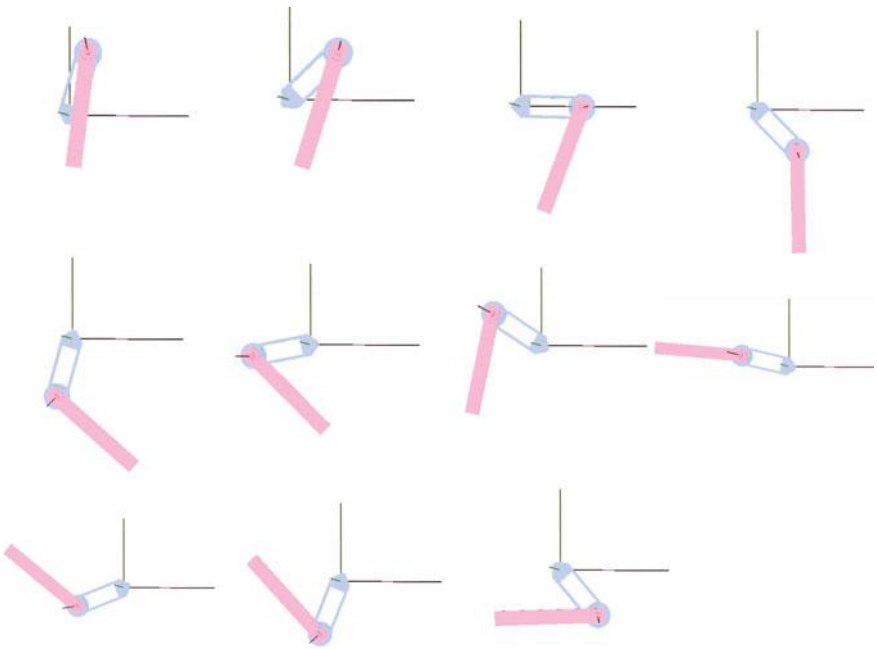


Fig. 11.6 Sequence of movement of a 2-DOF robot arm

$$V = \begin{pmatrix} C_1' \cdot \boldsymbol{L}_1' & 0 \\ C_2' \cdot \boldsymbol{L}_1' & C_2' \cdot \boldsymbol{L}_2' \end{pmatrix}, \tag{11.218}$$

$$\begin{aligned}
\dot{V}_{1,1} &= (C_1 \cdot \boldsymbol{L}_1) \cdot \boldsymbol{L}_1 \dot{q}_1 + (C_1 \cdot \boldsymbol{L}_2) \cdot \boldsymbol{L}_1 \dot{q}_2, \\
\dot{V}_{2,1} &= (C_2 \cdot \boldsymbol{L}_1) \cdot \boldsymbol{L}_1 \dot{q}_1 + (C_2 \cdot \boldsymbol{L}_2) \cdot \boldsymbol{L}_1 \dot{q}_2, \\
\dot{V}_{2,2} &= (C_2 \cdot \boldsymbol{L}_1) \cdot \boldsymbol{L}_2 \dot{q}_1 + (C_2 \cdot \boldsymbol{L}_2) \cdot \boldsymbol{L}_2 \dot{q}_2 + \frac{1}{2}(C_2 \cdot (\boldsymbol{L}_2 \boldsymbol{L}_1 - \boldsymbol{L}_1 \boldsymbol{L}_2))\dot{q}_1.
\end{aligned}$$

The matrix $\mathcal{M}$ is computed as

$$\begin{aligned}
\mathcal{M}_{1,1} &= m_1 l_{c1} l_{c1} + m_2 l_1 l_1 + m_2 l_{c2} l_{c2} + 2 m_2 l_1 l_{c2} \cos(q_2) + I_1 + I_2, \\
\mathcal{M}_{1,2} &= m_2 l_{c2} l_{c2} + m_2 l_1 l_{c2} \cos(q_2) + I_2, \\
\mathcal{M}_{2,2} &= m_2 l_{c2} l_{c2} + I_2.
\end{aligned}$$

And using conformal geometric algebra gives

$$\begin{aligned}
\mathcal{M}_{1,1} &= m_1 (V_{1,1} \cdot V_{1,1}) + m_2 (V_{2,1} \cdot V_{2,1}), \\
\mathcal{M}_{1,2} &= m_2 (V_{2,1} \cdot V_{2,2}), \\
\mathcal{M}_{2,2} &= m_2 (V_{2,2} \cdot V_{2,2}).
\end{aligned}$$

The element $M_{2,1} = M_{1,2}$ and M_I is computed as

$$M_I = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} I_1 & 0 \\ I_2 & I_2 \end{pmatrix} = \begin{pmatrix} I_1 + I_2 & I_2 \\ I_2 & I_2 \end{pmatrix}.$$

Similarly, the matrix G is computed using the equations

$$\begin{aligned}
G_1 &= (m_1 l_{c1} + m_2 l_1) g \sin(q_1) + m_2 g l_{c2} \sin(q_1 + q_2), \\
G_2 &= m_2 g l_{c2} \sin(q_1 + q_2).
\end{aligned}$$

In conformal geometry, they are given by

$$\begin{aligned}
G_1 &= m_1 (V_{1,1} \cdot g e_2) + m_2 (V_{2,1} \cdot g e_2), \\
G_2 &= m_2 (V_{2,2} \cdot g e_2).
\end{aligned}$$

Finally, the Coriolis matrix is given by

$$\begin{aligned}
C_{1,1} &= -m_2 l_1 l_{c2} \sin(q_2) \dot{q}_2, \\
C_{2,1} &= m_2 l_1 l_{c2} \sin(q_2) (\dot{q}_1), \\
C_{1,2} &= -m_2 l_1 l_{c2} \sin(q_2) (\dot{q}_1 + \dot{q}_2),
\end{aligned}$$

and in geometric algebra as

$$
\begin{aligned}
C_{1,1} &= m_1(V_{1,1} \cdot \dot{V}_{1,1}) + m_2(V_{1,2} \cdot \dot{V}_{2,1}),\\
C_{2,1} &= m_2(V_{2,2} \cdot \dot{V}_{2,1}),\\
C_{1,2} &= m_2(V_{1,2} \cdot \dot{V}_{2,2}),
\end{aligned}
$$

where $C_{2,2} = 0$. This is a toy example, but when the number of degrees of freedom increases, in contrast to the classical approach, the new one preserves its clarity.

Example 2 In this example, we show the form of the matrices for a six-degrees-of-freedom serial robot like a robot arm adept six 600 (see Fig. 11.7).

Matrix V

$$
V = \begin{pmatrix}
C_1' \cdot L_1' & 0 & 0 & 0 & 0 & 0\\
C_2' \cdot L_1' & C_2' \cdot L_2' & 0 & 0 & 0 & 0\\
C_3' \cdot L_1' & C_3' \cdot L_2' & C_3' \cdot L_3' & 0 & 0 & 0\\
C_4' \cdot L_1' & C_4' \cdot L_2' & C_4' \cdot L_3' & C_4' \cdot L_4' & 0 & 0\\
C_5' \cdot L_1' & C_5' \cdot L_2' & C_5' \cdot L_3' & C_5' \cdot L_4' & C_5' \cdot L_5' & 0\\
C_6' \cdot L_1' & C_6' \cdot L_2' & C_6' \cdot L_3' & C_6' \cdot L_4' & C_6' \cdot L_5' & C_6' \cdot L_6'
\end{pmatrix}, \tag{11.219}
$$

Matrix m

$$
m = \begin{pmatrix}
m_1 & 0 & 0 & 0 & 0 & 0\\
0 & m_2 & 0 & 0 & 0 & 0\\
0 & 0 & m_3 & 0 & 0 & 0\\
0 & 0 & 0 & m_4 & 0 & 0\\
0 & 0 & 0 & 0 & m_5 & 0\\
0 & 0 & 0 & 0 & 0 & m_6
\end{pmatrix}, \tag{11.220}
$$

Matrix a

$$
a = \begin{pmatrix}
ge_2\\
ge_2\\
ge_2\\
ge_2\\
ge_2\\
ge_2
\end{pmatrix}. \tag{11.221}
$$

Fig. 11.7 Sequence of movement of a 6-DOF robot arm

11.5.4 Computing $\mathcal{M}$

The component of kinetic energy $K_v = V^T m V$

$$\mathcal{M}_v = \begin{pmatrix} V_{1,1} & V_{2,1} & \cdots & V_{n,1} \\ 0 & V_{2,2} & \cdots & V_{n,2} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & V_{n,n} \end{pmatrix} \begin{pmatrix} m_1 & 0 & \cdots & 0 \\ 0 & m_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & m_n \end{pmatrix} \begin{pmatrix} V_{1,1} & 0 & \cdots & 0 \\ V_{2,1} & V_{2,2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ V_{n,1} & V_{n,2} & \cdots & V_{n,n} \end{pmatrix}. \tag{11.222}$$

The number of computations for mV is $\leq 3n(n+1)/2$ and the product of two matrices is

$$\mathcal{M}_v = \begin{pmatrix} V_{1,1} & V_{2,1} & \cdots & V_{n,1} \\ 0 & V_{2,2} & \cdots & V_{n,2} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & V_{n,n} \end{pmatrix} \begin{pmatrix} m_1 V_{1,1} & 0 & \cdots & 0 \\ m_2 V_{2,1} & m_2 V_{2,2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ m_n V_{n,1} & m_n V_{n,2} & \cdots & m_n V_{n,n} \end{pmatrix}. \tag{11.223}$$

The number of products for each component is given by

$$\begin{pmatrix} n & n-1 & n-2 & \cdots & 1 \\ n-1 & n-1 & n-2 & \cdots & 1 \\ n-2 & n-2 & n-2 & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots & \\ 1 & 1 & 1 & \cdots & 1 \end{pmatrix}. \tag{11.224}$$

The total number of computations is less than or equal to $\frac{1}{3}n^3 + \frac{1}{2}n^2 + \frac{1}{6}n$.

11.5.5 Computing G

$$G = V^T F. \tag{11.225}$$

The matrix G is computed as

$$G = \begin{pmatrix} V_{1,1} & V_{2,1} & \cdots & V_{n,1} \\ 0 & V_{2,2} & \cdots & V_{n,2} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & V_{n,n} \end{pmatrix} \begin{pmatrix} F_1 \\ F_2 \\ \vdots \\ F_n \end{pmatrix}. \tag{11.226}$$

The number of products for each component is given by

$$\begin{pmatrix} n \\ n-1 \\ n-2 \\ \vdots \\ 1 \end{pmatrix}. \tag{11.227}$$

The sum of $1 + 2 + 3 + \cdots + n = n(n+1)/2$. Then the total number of products needed to compute G is at most $\leq n(n+1)/2$.

11.6 Constrained Variational Problems

The essential idea of the use of Lagrange multipliers in the theory of optimization is to convert the constrained optimization problem to an unconstrained one. The procedure is as follows: form the *Lagrangian function*

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}) := f(\boldsymbol{x}) + \sum_{k=1}^{m} \lambda_k g_k(\boldsymbol{x}), \tag{11.228}$$

where the parameter variables $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_m)$ are called the Lagrange multipliers. Here $\mathcal{L}$ is a function of both $\boldsymbol{x}$ and $\boldsymbol{\lambda}$ involving $N + m$ variables. The partial derivatives of $\mathcal{L}$ w.r.t. all of its dependent variables are

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial x_j} &= \frac{\partial f}{\partial x_j} + \sum_{k=1}^{m} \lambda_k \frac{\partial g_k}{\partial x_j}, \\ \frac{\partial \mathcal{L}}{\partial \lambda_k} &= g_k, \end{aligned}$$

where $j = 1, \ldots, N$ and $k = 1, \ldots, m$. Note that at the stationary points of $\mathcal{L}(\boldsymbol{x}, \lambda)$, it is necessary that all these partial derivatives must be zero, in addition one must solve the following unconstrained problem:

$$\begin{aligned} \frac{\partial f}{\partial x_j} + \sum_{k=1}^{m} \lambda_k \frac{\partial g_k}{\partial x_j} &= 0, \\ g_k &= 0, \end{aligned}$$

where again $j = 1, \ldots, N$ and $k = 1, \ldots, m$. This equation system is solvable, as one has $N + m$ equations and also $N + m$ unknowns. Now, consider that one can solve this equation system to find a stationary point $\boldsymbol{x}^\star$, $\lambda^\star$ of $\mathcal{L}$ which could be one or more, then one concludes that $\boldsymbol{x}^\star$ is also a stationary point for the original constrained problem. Note that the goal of the use of the constraints on the constraint manifold

guarantees that one fulfills $\mathcal{L} = f + 0$, and that means strictly the stationary points of f and $\mathcal{L}$ coincide.

Given a Lagrangian $L(q, \dot{q}, t)$ for a system, suppose we realize the system has some constraints, which means that the q_j are not all independent. Furthermore, suppose that we have m holonomic constraints of the following form:

$$g_k(q_1, \ldots, q_n, t) = 0, \tag{11.229}$$

for $k = 1, \ldots, m < n$. Now, we can use the well-known method of Lagrange multipliers with Hamilton's principle to derive the equations of motion as follows:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_j}\right) - \frac{\partial L}{\partial q_j} = \sum_{k=1}^{n} \lambda_k(t) \frac{\partial g_k}{\partial q_j}, \tag{11.230}$$

for $j = 1, \ldots, n$ and $k = 1, \ldots, m$. The quantities on the right part

$$\sum_{k=1}^{n} \lambda_k(t) \frac{\partial g_k}{\partial q_j} \tag{11.231}$$

are called the generalized forces of constraint.

Note that there are mechanical systems with some types of non-holonomic constraints, in particular, constraints of the form

$$\sum_{j=1}^{n} C(\boldsymbol{q}, t)_{kj} \dot{q}_j + d_k(\boldsymbol{q}, t) = 0. \tag{11.232}$$

Here the assumption is that these equations are not integrable, in particular, not exact, otherwise the constraints would be holonomic.

11.6.1 Optimization

A common optimization problem is to extremize a functional J with respect to paths $\gamma = [a, b]$ which are constrained in some way. The task is to find the multidimensional curve $\boldsymbol{y} = \boldsymbol{y}(\boldsymbol{x}) \in \mathbb{R}^N$ and its derivative $\boldsymbol{y}_x \in \mathbb{R}^N$

$$\boldsymbol{y} = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_N \end{pmatrix} \qquad \boldsymbol{y}_x = \begin{pmatrix} y_1' \\ y_2' \\ y_3' \\ \vdots \\ y_N' \end{pmatrix} \tag{11.233}$$

($' \equiv \frac{\mathrm{d}}{\mathrm{d}x}$) that extremizes the functional

$$J(y) := \int_a^b F(x, \boldsymbol{y}, \boldsymbol{y}_x)dx, \tag{11.234}$$

subject to the set of m constraints, for $k = 1, \ldots, m < N$:

$$G_k(x, y) = 0 \tag{11.235}$$

being $\boldsymbol{y}(\boldsymbol{a})$ and $\boldsymbol{y}(\boldsymbol{a})$ fixed.

In order to solve this constrained variational problem, we will generalize the method of Lagrange multipliers. For that the m constraint equations from above imply

$$\int_a^b \lambda_k(x)G_k(x, \boldsymbol{y})dx = 0, \tag{11.236}$$

for each $k = 1, \ldots, m$, where the λ_k are the Lagrange multipliers, which with the constraints expressed in this integral form can, in general, be functions of x. Next, we formulate the equivalent of the Lagrangian function which corresponds the following functional:

$$\begin{aligned}\hat{J}(\boldsymbol{y}, \boldsymbol{\lambda}) &:= \int_a^b F(x, \boldsymbol{y}, \boldsymbol{y}_x)dx + \sum_{k=1}^m \int_a^b \lambda_k(x)G_k(x, \boldsymbol{y})dx, \\ &= \int_a^b (F(x, \boldsymbol{y}, \boldsymbol{y}_x) + \sum_{k=1}^m \lambda_k(x)G_k(x, \boldsymbol{y}))dx,\end{aligned} \tag{11.237}$$

where $\boldsymbol{\lambda} = \boldsymbol{\lambda}(x)$ is a vector of m Lagrange multiplier functions, i.e., $\boldsymbol{\lambda} = [\lambda_1, \lambda_2, \ldots, \lambda_m]^T$. The integrand of the functional above is

$$\hat{F}(x, \boldsymbol{y}, \boldsymbol{y}_x, \boldsymbol{\lambda}) + \sum_{k=1}^m \lambda_k(x)G_k(x, \boldsymbol{y}). \tag{11.238}$$

Now, we treat this as an unconstrained variational problem for the curves $\boldsymbol{y} = \boldsymbol{y}(x)$ and $\boldsymbol{\lambda} = \boldsymbol{\lambda}(x)$. From the classical variational problem if $(\boldsymbol{y}, \boldsymbol{\lambda})$ extremize J then necessarily they must satisfy the Euler–Lagrange equations, namely,

$$\frac{\partial \hat{F}}{\partial y_i} - \frac{d}{dx}\left(\frac{\partial \hat{F}}{\partial y_i'}\right) = 0, \tag{11.239}$$

$$\frac{\partial \hat{F}}{\partial \lambda_k} - \frac{d}{dx}\left(\frac{\partial \hat{F}}{\partial \lambda_k'}\right) = 0. \tag{11.240}$$

Note that $\hat{F} = \hat{F}(x, \boldsymbol{y}, \boldsymbol{y}_x, \boldsymbol{\lambda})$ is independent of explicit $\boldsymbol{\lambda}'$ so $\partial \hat{F}/\partial \lambda_k = 0$. Note that $\frac{\partial \hat{F}}{\partial \lambda_k} = G_k$, and hence the Euler–Lagrange equations above simplify to

$$\begin{aligned} \frac{\partial \hat{F}}{\partial y_i} - \frac{d}{dx}\left(\frac{\partial \hat{F}}{\partial y_i'}\right) &= 0, \\ G_k(x, \boldsymbol{y}) &= 0, \end{aligned} \tag{11.241}$$

for $i = 1, \ldots, N$ and $k = 1, \ldots, m$. This is a system of differential-algebraic equations: the first set of relations are nonlinear second-order ordinary differential equations, while the constraints are algebraic relations.

If the constraints are already in integral form, we have

$$\int_b^a G_k(x, \boldsymbol{y}) = 0, \tag{11.242}$$

for $k = 1, \ldots, m$, then we write,

$$\hat{J}(\boldsymbol{y}, \lambda) := J(\boldsymbol{y}) + \sum_{k=1}^{m} \lambda_k \int_b^a G_k(x, \boldsymbol{y}) dx, \tag{11.243}$$

$$= \int_a^b \big(F(x, \boldsymbol{y}, \boldsymbol{y}_x) + \sum_{k=1}^{m} \lambda_k G_k(x, \boldsymbol{y}) dx. \tag{11.244}$$

We have obtained a variational problem with respect to the curve $\boldsymbol{y} = \boldsymbol{y}(x)$ but the constraints are classical in the sense that the Lagrangian multipliers λ here are simply variables (and not functions). Proceeding as above, with this hybrid variational constraint problem, with

$$\hat{F}(x, \boldsymbol{y}, \boldsymbol{y}_x, \lambda) := F(x, \boldsymbol{y}, \boldsymbol{y}_x) + \sum_{k=1}^{m} \lambda_k \int_b^a G_k(x, \boldsymbol{y}) \tag{11.245}$$

we generate the Euler–Lagrange equations

$$\frac{\partial \hat{F}}{\partial y_i} - \frac{d}{dx}\left(\frac{\partial \hat{F}}{\partial y_i'}\right) = 0 \tag{11.246}$$

together with the integral constraint equations which result from taking the partial derivative of $\hat{J}(\boldsymbol{y}, \lambda)$ with respect to all the components of λ.

11.7 Euler–Lagrange Multivector Equations

We consider mechanical systems with holonomic constraints and all other forces conservative. Recall, we define the Lagrange function or Lagrangian is defined as

$$L = K - U, \tag{11.247}$$

where

$$K = \sum_{i=1}^{N} \frac{1}{2} m_i \parallel \dot{X}_i \parallel^2 \tag{11.248}$$

is the total kinetic energy for the system and V is its potential energy.

Euler's equation is a second-order differential equation which is an extremal of a function. It is used to study the principle of minimum action of a given system. Using a more general formulation, suppose that the system is described by the multivector variables X_i, $i = 1 \ldots n$. The Lagrangian L is a scalar-valued function dependent on X_i and $\dot{X}_i$ (derivative w.r.t. time t), and the time t. The action of the system is given by the functional

$$\mathcal{A} = \int_{t_1}^{t_2} L(X_i(t), \dot{X}_i(t), t) dt. \tag{11.249}$$

The principle of minimum action leads to Hamilton's principle which postulates that the integral of the Lagrangian w.r.t. time must be the minimum possible.

In order to solve this problem consider a function $L(X_i, \dot{X}_i, t)$ which varies in the interval $[t_1, t_2]$; to show the variation of the function $L(X_i, \dot{X}_i, t)$ which depends of t, one utilizes a δ for the contour conditions

$$X_i(t_1) = Y_{1i}, \quad X_i(t_2) = Y_{2i}. \tag{11.250}$$

Thus, a variation with $\delta X_i(t_1) = \delta X_i(t_2) = 0$ causes a functional variation given by

$$\delta \mathcal{A} = \int_{t_1}^{t_2} \delta L(X_i(t), \dot{X}_i(t), t) dt = \int_{t_1}^{t_2} \sum_{i=1}^{n} \left(\frac{\partial L}{\partial X_i} \delta X_i + \frac{\partial L}{\partial \dot{X}_i} \delta \dot{X}_i \right) dt. \tag{11.251}$$

Integrating by parts an expecting null variation at the extremes due to the contour conditions, we get

$$\delta \mathcal{A} = \int_{t_1}^{t_2} dt \sum_{i=1}^{n} \left(\frac{\partial L}{\partial X_i} - \frac{d}{dt} \left(\frac{\partial L}{\partial \dot{X}_i} \right) \right) \delta X_i. \tag{11.252}$$

Since the integral must equal zero for all the paths X_i, $i = 1 \ldots n$, we can gain the Euler–Lagrange equations as follows:

$$\frac{\partial L}{\partial X_i} - \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{X}_i}\right) = 0, \quad \forall i =, \ldots, n. \tag{11.253}$$

Note that the multivector derivative guarantees that there are as many equations as the grades of X_i, which means that the number of obtained equations equals as the degrees of freedom of the system.

11.8 Hamiltonian Mechanics on Phase Space

In this section, we will study the Hamiltonian mechanics formalism and the Poisson bracket in the phase space. We will follow the equations proposed by Abou El Dahab [5] and Pappas [6] and the axioms of [7–9].

11.8.1 Symplectic Structure on Vector Manifolds

Consider that a vector manifold $\mathcal{M}$ is a set whose points are assumed to be vectors on which a geometric algebra G_n is defined. The difference of two vectors is called a chord, and the limit of a sequence of chords defines a tangent vector. The set of all vector tangents to $\mathcal{M}$ at a point x is the tangent space at x. This tangent space defines itself as another related geometric algebra called the tangent algebra of $\mathcal{M}$ at x. Note that the projection of any multivector X into the tangent algebra at x is $P(X) = [X \cdot I(x)] \cdot I^{-1}(x)$, where $I(x)$ is the pseudoscalar of the tangent algebra. If $P(X) = X$, then X is tangent to $\mathcal{M}$ at x. A multivector function which is tangent to $\mathcal{M}$ at every point $x \in \mathcal{M}$ is called a field on $\mathcal{M}$. Given $\mathcal{M}^{2m}$ be the phase of a physical system; each vector x represents a state of the system as we represent the whole system in state variables. A bivector field $\boldsymbol{G}(x)$ determines a skew-symmetric bilinear form f on $\mathcal{M}^{2m}$, i.e.,

$$g(x, y) = x \cdot (y \cdot \boldsymbol{G}) = (x \wedge y) \cdot \boldsymbol{G}. \tag{11.254}$$

Here the bivector field $\boldsymbol{G}$ plays the role of a closed non-degenerated differential two-form $w = \sum dp_i \wedge dq_i$ in defining a symplectic structure.

The analytical version of the Hamiltonian mechanics formalism studies the Hamilton's equations,

$$\dot{q}_i = \frac{\partial H}{\partial p_i}, \quad \dot{p}_i = -\frac{\partial H}{\partial q_i}, \tag{11.255}$$

as first-order differential equations, which yield integrals by using the invariance of the equations under canonical transformations.

Next, we will outline the symplectic version of the Hamilton's formalism. Considering the vector manifold $\mathcal{M}$, the Hamiltonian $H(x)$ determines a gradient system expressed by the following autonomous differential equations:

$$\dot{x} = \partial H \cdot \boldsymbol{G}, \tag{11.256}$$

where the overdot indicates the derivative w.r.t. time. Note that for scalar fields $\partial f = \nabla f$, so $\dot{x} = \nabla H \cdot \boldsymbol{G}$.

If $x = x(t)$ is an integral curve of Eq. (11.256), and $f(x)$ is any function, similarly as Eq. (11.254), one can easily show that

$$\dot{f} = (\partial f \wedge \partial H) \cdot \boldsymbol{G}, \tag{11.257}$$

henceforth, $\dot{H} = 0$, which tells us that the system conserves energy and it is not dissipative.

In order to show that Eq. (11.256) represents Hamiltonian's equations, first consider $\mathcal{M}^{2m} = \mathbb{R}^{2m}$, then the bivector $\boldsymbol{G}$ can be decomposed into a sum

$$\boldsymbol{G} = \sum_i = e_{p_i} \wedge e_{q_i}, \tag{11.258}$$

where the set of vectors $\{e_p\}$ and $\{e_q\}$ are orthonormal and mutually orthogonal. If we write

$$\partial H = \sum_i \Big(\frac{\partial H}{\partial q_i} e_{q_i} + \frac{\partial H}{\partial p_i} e_{p_i}\Big), \tag{11.259}$$

using Eq. (11.255), the gradient system of Eq. (11.256) is then rewritten as follows:

$$\dot{x} = \partial H \cdot \boldsymbol{G} = \sum_i \Big(\frac{\partial H}{\partial q_i} e_{q_i} + \frac{\partial H}{\partial p_i} e_{p_i}\Big) \cdot \boldsymbol{G} = \sum_i \big(- \dot{p}_i e_{p_i} + \dot{q}_i e_{q_i}\big). \tag{11.260}$$

11.8.2 The Hamiltonian and the Poisson Bracket

The Hamiltonian mechanics is described by the coordinates $\{q_1, q_2, \ldots, q_n\}$, let it firstly define a configuration space as n-D real vector space $\mathbb{R}^N$ which is spanned by an orthonormal basis $\{e_n\}$, with the properties

$$e_\mu \cdot e_\nu = \frac{1}{2}\{e_\mu, e_\nu\} = \frac{1}{2}(e_\mu e_\nu + e_\nu e_\mu) = \delta_{\mu\nu}, \tag{11.261}$$

for $\mu, \nu = 1, 2, \ldots, n$. The state of an electromechanical or robot system can be represented by the vectors

$$q = q^{\mu} e_{\mu}. \tag{11.262}$$

Such vectors in configuration space generate the geometric algebra G_n, with geometric product

$$q_a q_b = q_a \cdot q_b + q_a \wedge q_b. \tag{11.263}$$

The vector derivative ∂_q specified in terms related to the coordinates reads

$$\partial_q = \frac{\partial}{\partial_q^{\mu}} e^{\mu}, \tag{11.264}$$

where e^{μ} is the reciprocal basis for $\mathbb{R}^N$.

In similar way, we define the momentum space as a n-D real vector space $\bar{\mathbb{R}}^N$ which is spanned by an orthonormal basis $\{\bar{e}_n\}$, with the properties

$$\bar{e}_{\mu} \cdot \bar{e}_{\nu} = \frac{1}{2}\{\bar{e}_{\mu}, \bar{e}_{\nu}\} = \frac{1}{2}(\bar{e}_{\mu}\bar{e}_{\nu} + \bar{e}_{\nu}\bar{e}_{\mu}) = \delta_{\mu\nu}, \tag{11.265}$$

for $\mu, \nu = 1, 2, \ldots, n$. Henceforth, the momentum of an electromechanical or robot system can be expressed as a vector as follows:

$$\bar{p} = p^{\mu} \bar{e}_{\mu}. \tag{11.266}$$

The derivative with respect to $\bar{p}$ reads

$$\partial_{\bar{p}} = \frac{\partial}{\partial_{\bar{p}}^{\mu}} \bar{e}^{\mu}, \tag{11.267}$$

where $\bar{e}^{\mu}$ is the reciprocal basis for $\bar{\mathbb{R}}^N$. Thus, the symplectic phase space $\mathbb{R}^{2n}$ is represented as a direct sum of the $\mathbb{R}^n$ and $\bar{\mathbb{R}}^n$ spaces, namely,

$$\mathbb{R}^{2n} = \mathbb{R}^n \otimes \bar{\mathbb{R}}^n. \tag{11.268}$$

This symplectic 2n-D space has the geometric algebra $G_{n,n}$, which is completely defined by Eqs. (11.261–11.265) with the following orthogonal relations:

$$e_{\mu} \cdot \bar{e}_{\nu} = \frac{1}{2}\{e_{\mu}, \bar{e}_{\nu}\} = \frac{1}{2}(e_{\mu}\bar{e}_{\nu} + \bar{e}_{\nu}e_{\mu}) = 0. \tag{11.269}$$

Introducing the symplectic bivector [8, 10]

$$J = \sum_{\mu} J_{\mu}, \tag{11.270}$$

where the components are two blades

$$J_{\mu} = e_{\mu}\bar{e}_{\mu} = e_{\mu} \wedge \bar{e}_{\mu}. \tag{11.271}$$

One can prove that

$$\begin{aligned} e_{\mu} \cdot J &= \bar{e}_{\mu}, \\ \bar{e}_{\mu} \cdot J &= -e_{\mu}. \end{aligned} \tag{11.272}$$

Additionally,

$$J_{\mu}^2 = -1, \qquad J^2 = -n, \tag{11.273}$$

where $\frac{J}{\sqrt{n}}$ represents the unit imaginary i, see [8].

In order to define the Hamiltonian mechanics on the phase space [5], one can consider the state of the physical system by a single point θ in the phase space which is defined as

$$\theta = q + \bar{p} = q^{\mu} e_{\mu} + p^{\mu} \bar{e}_{\mu}. \tag{11.274}$$

Let us define

$$\theta^r = \begin{cases} q^r & 1 \leq r \leq n \\ p^{r-n} & n < r \leq 2n, \end{cases} \tag{11.275}$$

$$E_r = \begin{cases} e^r & 1 \leq r \leq n \\ \bar{e}_{r-n} & n < r \leq 2n. \end{cases} \tag{11.276}$$

Equation (11.274) can be then rewritten as follows:

$$\theta = \theta^r E_r. \tag{11.277}$$

Using Eq. (11.278), $\bar{E}_r$ can be expressed as

$$\bar{E}_r = E_r \cdot J = \begin{cases} \bar{e}_r & 1 \leq r \leq n \\ -e_{r-n} & n < r \leq 2n. \end{cases} \tag{11.278}$$

One can define then the scalar product in this symplectic space as follows:

$$(E_r, E_s)^s = E_r \bar{E}_s = \frac{1}{2}\{E_r, \bar{E}\}, \tag{11.279}$$

which written in a matricial form reads

$$(E_r, E_s)^s = \gamma_{rs} = \begin{pmatrix} 0 & -\delta_{\mu\nu} \\ \delta_{\mu\nu} & 0 \end{pmatrix}. \tag{11.280}$$

One can now derive the derivative with respect to a phase point θ as follows:

$$\partial_\theta = \partial_q + \partial_{\bar{p}} = \frac{\partial}{\partial q^\mu} e_\mu + \frac{\partial}{\partial p^\mu} \bar{e}_\mu = \frac{\partial}{\partial \theta^r} E^r = \partial_\theta \cdot E, \tag{11.281}$$

similarly with respect to a phase point $\bar{\theta}$ as follows:

$$\partial_{\bar{\theta}} = \partial_q + \partial_{\bar{p}} = \frac{\partial}{\partial q^\mu} \bar{e}_\mu - \frac{\partial}{\partial p^\mu} e_\mu = \frac{\partial}{\partial \theta_r} J = \partial_\theta \cdot J. \tag{11.282}$$

Hamiltonian mechanics was first formulated by William Rowan Hamilton in 1833, starting from Lagrangian mechanics, a previous reformulation of classical mechanics introduced by Joseph Louis Lagrange in 1788. The Hamiltonian of an electromechanical or robotic system is a scalar-valued function

$$H(q, \dot{q}) = K(q, \dot{q}) - L(q, \dot{q}), \tag{11.283}$$

where q stands for positional coordinates and $\dot{q}$ stands for generalized velocities or both are called the canonical coordinates. On the other hand, the Hamiltionian can be formulated as an explicit function of the positional coordinates and the momenta

$$H(p, \dot{q}) = K(p, \dot{q}) - L(p, \dot{q}). \tag{11.284}$$

The momenta are computed as the derivatives of the Lagrangian, namely,

$$p_i = \frac{\partial L(p_i, \dot{q}_i)}{\partial \dot{q}_i}, \quad \dot{q}_i = \frac{\partial L(p_i, \dot{q}_i)}{\partial p_i}. \tag{11.285}$$

Thus, we are now in the condition to formulate the Hamiltionian from the Lagrangian, only if it also fulfills the required condition of the theorem of an implicit function. In his regard, the Hamiltonian can be formulated as Legendre's transformation of the Lagrangian. The Legrende transform replaces the velocities by momenta and puts the equation in Hamiltonian form as follows:

$$H = H(q, p, t) = \sum_i \dot{q}^i p_i - L(p_i, \dot{q}^i, t), \tag{11.286}$$

where

$$\dot{q}^i = \frac{\partial H}{\partial p_i}, \quad p^i = -\frac{\partial H}{\partial \dot{q}^i}. \tag{11.287}$$

Here q, p stand as the coordinates for position and momentum.[1]

In order to show that the Hamiltonian for an autonomous system is a motion constant, we take the total time derivative of Eq. (11.286) as follows:

$$\frac{dH}{dt} = \sum \left(\dot{q}^i \dot{p}_i + \ddot{q}^i p_i - \frac{\partial L}{\partial q_i} \dot{q}_i - \frac{\partial L}{\partial \dot{q}_i} \ddot{q}_i \right) - \frac{\partial L}{\partial t}. \tag{11.288}$$

Given $H(q, p, t)$, the total differential of H is given by

$$\begin{aligned} dH &= \sum_i \left(\frac{\partial H}{\partial q^i} dq^i + \frac{\partial H}{\partial p_i} dp_i \right) + \frac{\partial H}{\partial t} dt, \\ &= \sum_i \left(\dot{q}^i dp_i + p_i d\dot{q}^i - \frac{\partial L}{\partial q^i} dq^i - \frac{\partial L}{\partial \dot{q}^i} d\dot{q}^i \right) - \frac{\partial L}{\partial t} dt \end{aligned} \tag{11.289}$$

or differentiating the Eq. (11.286),

$$dH = \sum_i \left(\dot{q}^i dp_i - \dot{p}_i dq^i \right) - \frac{\partial L}{\partial t} dt. \tag{11.290}$$

If we equal the last two equations

$$\begin{aligned} dH &= \sum_i \left(\frac{\partial H}{\partial q^i} dq^i + \frac{\partial H}{\partial p_i} dp_i \right) + \frac{\partial H}{\partial t} dt, \\ &= \sum_i \left(\dot{q}^i dp_i - \dot{p}_i dq^i \right) - \frac{\partial L}{\partial t} dt, \end{aligned} \tag{11.291}$$

one can then identify the multiplicative elements of dq^i and dp_i to obtain the following relations:

$$\dot{p}_i = -\frac{\partial H}{\partial q^j}, \quad \dot{q}^i = \frac{\partial H}{\partial p_i}, \quad \frac{\partial H}{\partial t} = -\frac{\partial L}{\partial t}, \tag{11.292}$$

or

$$\frac{dH}{dt} = \frac{\partial H}{\partial t}. \tag{11.293}$$

This result shows that if H doesn't depend explicitly on the time then $\dot{H}$=0, which means that the Hamiltonian H conserves itself or that the system is autonomous or a conservative system.

[1] In mathematics, two differentiable functions f and g are related via a Legrende transformation if each of their first derivatives is inverse function of the other, i.e., $Df = (Dg)^{-1}$.

Hamilton's equation for a phase space trajectory, $\theta = \theta(t)$, of the system takes the following form:

$$\dot{\theta} = \partial_{\bar{\theta}} H, \tag{11.294}$$

such equation can be written in its conventional form as Eq. (11.255)

$$\dot{q}^{\mu} e_{\mu} + \dot{p}^{\mu} \bar{e}_{\mu} = \Big(\frac{\partial H}{\partial p^{\mu}} e_{\mu} - \frac{\partial H}{\partial q^{\mu}} \bar{e}_{\mu} \Big), \tag{11.295}$$

where according to Eq. (11.287)

$$\dot{q}^{\mu} = \frac{\partial H}{\partial p^{\mu}}, \quad \dot{p}^{\mu} = -\frac{\partial H}{\partial q^{\mu}}. \tag{11.296}$$

We still need an extra definition to be able to represent the Hamiltonian of Eq. (11.294), and this is the standard expression for the Poisson bracket. Given two scalar-valued functions, $f(\theta)$, $g(\theta)$, the Poisson bracket reads

$$\{f, g\} = (\partial f \wedge \partial g) \cdot \boldsymbol{G} = \partial f \cdot (\partial g \cdot \boldsymbol{G}) = \partial_{\bar{\theta}} f \cdot \partial_{\theta} g, \tag{11.297}$$

which can be rewritten using Eqs. (11.261), (11.265), (11.269), (11.281), and (11.282) as follows:

$$\{f, g\} = \frac{\partial f}{\partial q^{\mu}} \frac{\partial g}{\partial p^{\mu}} - \frac{\partial f}{\partial p^{\mu}} \frac{\partial g}{\partial q^{\mu}}, \tag{11.298}$$

as you can recognize is the conventional definition in terms of the phase space coordinates.

Henceforth, for any multivector-valued function $M = M(\theta)$, in geometric algebra its equation of motion using Eqs. (11.261), (11.265), (11.269), (11.274), (11.281), (11.282), and (11.294) can be written as

$$\dot{M} = \dot{\theta} \cdot \partial_{\theta} M = \partial_{\bar{\theta}} H \cdot \partial_{\theta} M = \{H, M\}. \tag{11.299}$$

For $M = \theta$, the Hamilton equation (11.294) can be formulated as follows:

$$\dot{\theta} = \{H, \theta\}. \tag{11.300}$$

As a final remark, it can be shown that the equations of motion for q_i and p_i, Hamilton's equations (11.287), are special cases of this. Moreover, the Poisson brackets for the phase space variables q_i and p_i are

$$\{q^i, q^j\} = 0,$$
$$\{q^i, p_j\} = \delta^i_j$$
$$\{p_i, p_j\} = 0.$$

According to [11], this type of formal manipulations does not have much relevance to actual problem—solving in classical mechanics. However, it is very useful, when we study the behavior of some given system in field or particle theory. It is also a powerful formulation to handle systems with constraints. It is here mentioned, because it shows its straightforward relation with quantum mechanics. A way to go from a classical to a quantum system, it is to replace the Poisson bracket by (i) times the commutator $[A, B] = AB - BA$. This means that one has $qp - pq = i$, note that position and momentum no longer commute. The momentum can actually be represented as a space derivative, $p = i\frac{\partial}{\partial q}$. One sees that the two variables play a role of *operators*, and their values cannot simultaneously be given specific values, because ordinary numbers commute. This leads certainly to Heisenberg's *uncertainty principle*, which states the impossibility to carry out measurements on both q and p simultaneously beyond a maximal precision.

11.9 Integral Invariants

For an optimization problem, we seek for the equations of the paths, parameterized curves γ_i within $[t_1, t_2]$ for which the action J is stationary. Given a vector field F on $\mathcal{M}^{2m}$ and γ a parameterized curve, then one can say that the integral,

$$J = \int_\gamma F \cdot d\boldsymbol{x}, \tag{11.301}$$

is an integral invariant of the vector field $\boldsymbol{X}$ if $\frac{dJ}{dt} = 0$ along the flow of $\boldsymbol{X}$, defined by $\dot{x} = \boldsymbol{X}$; that is to say , the value of J along a curve γ is the same as along the image of γ' under the flow of $\boldsymbol{X}$. In order to show the condition for J to be an integral invariant, one can proceed as follows: choose a parameter s for a fixed t and define γ in terms of $s \in [0, 1]$. For s fixed, the same point $\boldsymbol{x}$ is situated on the trajectory then $\boldsymbol{x} = \boldsymbol{x}(s, t)$ and $\frac{\partial \boldsymbol{x}}{\partial t} = \boldsymbol{X}$, henceforth one can rewrite the integral as follows:

$$J = \int_0^1 F \cdot \frac{\partial \boldsymbol{x}}{\partial s} ds. \tag{11.302}$$

Taking the derivative of J w.r.t. to t

$$\frac{dJ}{dt} = \int_0^1 \frac{\partial}{\partial t}\Big(F \cdot \frac{\partial \boldsymbol{x}}{\partial s}\Big)ds = \int_0^1 \Big(\frac{\partial F}{\partial t} \cdot \frac{\boldsymbol{x}}{\partial s} + F \cdot \frac{\partial \boldsymbol{X}}{\partial s}\Big)ds \tag{11.303}$$
$$= \int_0^1 [(\boldsymbol{X} \cdot \partial)F + \dot{\partial}(F \cdot \dot{\boldsymbol{X}})] \cdot \frac{\partial \boldsymbol{x}}{\partial s} ds = \int_\gamma [(\boldsymbol{X} \cdot \partial)F + \dot{\partial}(F \cdot \dot{\boldsymbol{X}})] \cdot d\boldsymbol{x},$$

where the overdot indicates the action of ∂ on the dotted variable. According to ([8], p. 53), we can formulate

$$\partial(F \cdot \boldsymbol{X}) = \boldsymbol{X} \cdot \partial F + F \cdot \partial \boldsymbol{X} - F \cdot (\partial \wedge \boldsymbol{X}) - \boldsymbol{X} \cdot (\partial \wedge F). \tag{11.304}$$

Since

$$\dot{\partial}(F \cdot \dot{\boldsymbol{X}}) = F \cdot \partial \boldsymbol{X} - F \cdot (\partial \wedge \boldsymbol{X}), \tag{11.305}$$

one can then write the integrand above as follows:

$$\begin{aligned} \boldsymbol{X} \cdot \partial F + \dot{\partial}(F \cdot \dot{\boldsymbol{X}}) &= \boldsymbol{X} \cdot \partial F + F \cdot \partial \boldsymbol{X} - F \cdot (\partial \wedge \boldsymbol{X}), \\ &= \partial(F \cdot \boldsymbol{X}) + \boldsymbol{X} \cdot (\partial \wedge F). \end{aligned} \tag{11.306}$$

Thus, Eq. (11.303) can be rewritten as

$$\frac{dJ}{dt} = F \cdot \boldsymbol{X}|_\gamma + \int_\gamma [\boldsymbol{X} \cdot (\partial \wedge F)] \cdot dx, \tag{11.307}$$

and concluding F is an integral invariant of $\boldsymbol{X}$ if and only if Eq. (11.307) is zero for every γ. For that sufficient the conditions are $F \cdot \boldsymbol{X} = 0$ and $\boldsymbol{X} \cdot (\partial \wedge F) = 0$. Let us discuss certain kinds of paths, if γ is closed, the first term of Eq. (11.307) becomes zero and we can claim that F is an (relative) integral invariant of F if and only if

$$\oint [\boldsymbol{X} \cdot (\partial \wedge F)] \cdot d\boldsymbol{x} = 0 \qquad \texttt{for every closed } \gamma, \tag{11.308}$$

or if and only if

$$\boldsymbol{X} \cdot (\partial \wedge F) = \partial h \qquad \texttt{for some function } h. \tag{11.309}$$

Let us consider a simple example, given the Hamiltonian $H = \frac{1}{2}(u^2 + v^2) + \frac{1}{2}y^2 \in \mathbb{R}^4$, then $F = ue_{p_1} + ve_{p_2}$ is a relative integral invariant of the field $\dot{\boldsymbol{x}} = \boldsymbol{X} = \nabla H \cdot \boldsymbol{G} = ue_{p_1} + ve_{p_2} - ye_{q_2}$, and this follows Eq. (11.309), namely,

$$\boldsymbol{X} \cdot (\nabla \wedge F) = -\nabla H. \tag{11.310}$$

According to ([8], p. 258), Stoke's theorem can be written in terms of the vector field F as

$$\int dV \cdot (\nabla \wedge F) = \oint dS \cdot F, \tag{11.311}$$

where dV and dS are direct volume elements on $\mathcal{M}^{2m}$ and $\partial\mathcal{M}^{2m}$, and the bivector field $\boldsymbol{G} = \sum_i e_{p_i} \wedge e_{q_i}$ defining the symplectic structure on $\mathcal{M}^{2m}$ is an absolute integral invariant of any Hamiltonian vector field.

11.9.1 Lagrangian and Hamiltonian for Electromechanical or Robot Systems

The Euler–Lagrange equation of an electromechanical or robot systems is given by [12]

$$I(\theta)\ddot{q} + C(q, \dot{q}) + G(q) = \tau, \tag{11.312}$$

where $I(\theta)$, $C(q, \dot{q})$, $G(q)$, and τ stand for the inertia matrix, Coriolis matrix, the gravity, and the torque, respectively.

Lagrange's equation for electromechanical or robot systems is given by

$$\frac{d\partial_{\dot{\theta}} L(\theta, \dot{\theta})}{dt} - \partial_\theta L(\theta, \dot{\theta}) = \tau, \tag{11.313}$$

where L is the Lagrangian of the system computed as the difference between the kinetic and potential energies as follows:

$$L(\theta, \dot{\theta}) = K(\theta, \dot{\theta}) - V(\theta). \tag{11.314}$$

The generalized momentum is defined as

$$p = \partial_{\dot{\theta}} L(\theta, \dot{\theta}). \tag{11.315}$$

Taking the derivative of the Hamiltonian in the phase space Eq. (11.287), we get

$$\partial_\theta H(\theta, p) = \partial_\theta \big(p \cdot \dot{\theta} - L(\theta, \dot{\theta})\big) = -\partial_\theta L(\theta, \dot{\theta}), \tag{11.316}$$

henceforth

$$\partial_\theta H(\theta, p) = -\partial_\theta L(\theta, \dot{\theta}). \tag{11.317}$$

Using Eqs. (11.313), (11.315), and (11.287), we obtain

$$\dot{p} - \partial_\theta L(\theta, \dot{\theta}) = \tau. \tag{11.318}$$

Now taking Eqs. (11.313), (11.317), and (11.318), the derivative of the momentum reads

$$\dot{p} = \tau - \partial_\theta H(\theta, p). \tag{11.319}$$

Similarly

$$\partial_p H(\theta, p) = \partial_p \big(p \cdot \dot{\theta} - L(\theta, \dot{\theta})\big) = \dot{\theta}, \tag{11.320}$$

so

$$\dot{\theta} = \partial_p H(\theta, p). \tag{11.321}$$

Equations (11.319) and (11.321) formulated in the geometric algebra framework correspond to Hamilton's equations (11.255).

In an autonomous system, it is possible to represent the Lagrangian equation and the Hamiltonian equation as follows:

$$L(\theta, \dot{\theta}) = K(\theta, \dot{\theta}) - V(\theta), \tag{11.322}$$
$$H(\theta, p) = K(\theta, p) + V(\theta). \tag{11.323}$$

From Eqs. (11.319) and (11.323), one obtains

$$\begin{aligned} \dot{p} &= \tau - \partial_\theta H(\theta, p) = \tau - \partial_\theta K(\theta, p) - \partial_\theta V(\theta), \\ &= \tau - \partial_\theta K(\theta, p) - G(\theta), \end{aligned} \tag{11.324}$$

$$p = \partial_{\dot{\theta}} L = I(\theta)\dot{\theta}, \tag{11.325}$$

where the velocity $\dot{\theta}$ is transformed affine to the momentum via the tensor inertia $I(\theta)$, and thus the velocity is

$$\dot{\theta} = I(\theta)^{-1} p. \tag{11.326}$$

The kinetic energy in terms of coordinates reads

$$K(\theta, \dot{\theta}) = \frac{1}{2}\dot{\theta}^T I(\theta)\dot{\theta}, \tag{11.327}$$

so using Eq. (11.325), it can be written in terms of the moment

$$K(\theta, p) = \frac{1}{2} p^T I(\theta)^{-1} p. \tag{11.328}$$

11.10 Iterative Computing of the Local Hamiltonian and the Derivative of the Momenta

The next equations are derived according to Eqs. (11.313–11.319). The local Lagrangian L_i of the robot manipulator at the i-joint can be computed as the difference between the kinetic and potential energies as follows:

$$L(\theta, \dot{\theta})_i = K(\theta, \dot{\theta})_i - V(\theta)_i. \tag{11.329}$$

The momenta p_i is defined as

$$p_i = \partial_{\dot{\theta}_i} L(\theta, \dot{\theta})_i. \tag{11.330}$$

The Hamiltonian of canonical coordinates at the i-joint i given by

$$H(q_i, p_i)_i = \sum_i \dot{q}_i p_i - L(q_i, \dot{q}^i). \tag{11.331}$$

Using Legendre's transformation, the Hamiltonian of canonical coordinates of Eq. (11.331) is transformed to the Hamiltonian in the phase space at the i-joint

$$H(\theta_i, p_i)_i = p_i \cdot \dot{\theta}_i - L(\theta_i, \dot{\theta}_i). \tag{11.332}$$

Taking the derivative of Eq. (11.332), we get

$$\partial_\theta H(\theta_i, p_i) = \partial_{\theta_i}\big(p_i \cdot \theta_i - L(\theta_i, \dot{\theta}_i)\big) = -\partial_{\theta_i} L(\theta_i, \dot{\theta}_i), \tag{11.333}$$

henceforth

$$\partial_{\theta_i} H(\theta_i, p_i) = -\partial_{\theta_i} L(\theta_i, \dot{\theta}_i). \tag{11.334}$$

Using Eqs. (11.313), (11.330), and (11.332), we obtain

$$\dot{p}_i - \partial_{\theta_i} L(\theta_i, \dot{\theta}_i) = \tau_i. \tag{11.335}$$

Now taking Eqs. (11.329), (11.334), and (11.318), the time derivative of the i-momentum reads

$$\dot{p}_i = \tau_i - \partial_{\theta_i} H(\theta_i, p_i). \tag{11.336}$$

Thus, using these last equations, we can compute iteratively locally at each joint of the robot manipulator the wrench $\boldsymbol{w}_i$ Eq. (11.117), torque τ_i Eq. (11.118), the Hamiltonian $H(\theta_i, p_i)$ Eq. (11.332), and the time derivative of the i-momentum $\dot{p}_i$

Eq. (11.336). We will use the computed time derivative of the i-momentum to derive a local controller in terms of a screw, and this is explained in Chap. 12.

Using this Newton–Euler algorithm of Sect. 11.4.4, we can now extend it to compute the local Hamiltonians $H(\theta_i, p_i)$ and the time derivative of the i-momentum $\dot{p}_i$ at each robot joint as follows:

Outward computing:

• Compute the link velocity or twist $\boldsymbol{s}_j$ at each link

$$\begin{aligned} \boldsymbol{s}_1^0 &= \dot{\theta}_1 \boldsymbol{L}_1^0, \\ \boldsymbol{s}_j^0 &= \widetilde{\boldsymbol{M}}_j \boldsymbol{s}_{j-1}^0 \boldsymbol{M}_j + \dot{\theta}_j \boldsymbol{L}_j^0. \end{aligned} \tag{11.337}$$

• Compute the link acceleration $\dot{\boldsymbol{s}}_j$ at each link

$$\begin{aligned} \dot{\boldsymbol{s}}_1^0 - \boldsymbol{G}_1^0 &= \ddot{\theta}_1 \boldsymbol{L}_1^0 - \boldsymbol{G}, \\ (\dot{\boldsymbol{s}}_j^0 - \boldsymbol{G}_j^0) &= \widetilde{\boldsymbol{M}}_j (\dot{\boldsymbol{s}}_{j-1}^0 - \boldsymbol{G}_{j-1}^0) \boldsymbol{M}_j + \ddot{\theta}_j \boldsymbol{L}_j^0 + \dot{\theta}_j \left[\boldsymbol{s}_j^0, \boldsymbol{L}_j^0 \right]. \end{aligned} \tag{11.338}$$

Inward computing:

• According to Eq. (11.332), compute the Hamiltonian H_j either in generalized coordinates or in velocity and momentum (phase space) at each link

$$L(\theta_i, \dot{\theta}_i) = K(\theta_i, \dot{\theta}_i) - V(\theta_i), \tag{11.339}$$

$$H(\theta_i, \boldsymbol{p}_i) = N_i(\boldsymbol{s}_i) \cdot \boldsymbol{s}_i - L(\theta_i, \dot{\theta}_i) = \boldsymbol{p}_i \cdot \boldsymbol{s}_i - L(\theta_i, \dot{\theta}_i). \tag{11.340}$$

• Compute the wrench $\boldsymbol{w}_j$ due to each link

$$\boldsymbol{w}_j^0 = N_j^0(\dot{\boldsymbol{s}}_j^0 - \boldsymbol{G}_j^0) + \{\boldsymbol{s}_j^0, N_j^0(\boldsymbol{s}_j^0)\}. \tag{11.341}$$

• Compute the total wrench acting $\boldsymbol{W}_i$ on each link

$$\begin{aligned} \boldsymbol{W}_K^0 &= \boldsymbol{w}_K^0, \\ \boldsymbol{W}_i^0 &= \widetilde{\boldsymbol{M}}_j \boldsymbol{W}_{j+1}^0 \boldsymbol{M}_j + \boldsymbol{w}_j^0. \end{aligned} \tag{11.342}$$

• Like Eq. (11.140), compute the torques at each link by pairing with the joint screws $\boldsymbol{L}_i^0$,

$$\tau_i = < \boldsymbol{W}_i^0 \boldsymbol{L}_i^0 >_0 = \boldsymbol{W}_i^0 \cdot \boldsymbol{L}_i^0. \tag{11.343}$$

• According to Eq. (11.335), compute the derivative of the momenta at each link

$$\dot{p}_i = \tau_i - \partial_{\theta_i} L(\theta_i, \dot{\theta}_i). \tag{11.344}$$

Note that the use of the Newton–Euler iterative algorithm makes easy to compute the local Hamiltonians of more complex robot systems as n-D robot arms or n-D inverted pendulums.

11.11 Conclusion

This chapter has shown the advantages of the use of geometric algebra framework to solve problems of dynamics of robot manipulators. The traditional approaches based on vector calculus, quaternions algebra, or linear algebra are complicate to handle the kinematics and dynamics involving points, lines, and planes. Conformal geometric algebra allows a more complete repertoire of geometric primitives and also it offers versors which are a much more efficient representations of linear transformations.

When you are dealing with Euler–Lagrange equations for robot dynamics, depending upon the degrees of freedom of the system, the equation will have big inertia and Coriolis tensors. How to treat those tensors within the geometric algebra?. One way is to disentangle those tensors in terms of matrices with entries as inner products between screw axes and the mass centers of the robot limbs. In this manner, we avoid quadratic entries in the tensors. A reformulation of these tensors in this way using conformal geometric algebra facilitates the plant identification, namely, we can measure the screw axes and carry out the inner products between these lines and the mass centers. After this study, you can see that it would be very difficult to factorize the tensors using matrix algebra. We hope that the reader thanks to this illustration he or she will be stimulated to apply conformal geometric algebra to develop new algorithms for system identification and for nonlinear control of robot mechanisms.

This work also presents the dynamic model and the nonlinear control for robot manipulators using the geometric algebra framework specifically utilizing the motor algebra $G^{+}_{3,0,1}$. In contrast to the use of Euler–Lagrange equations, the dynamic model of a robot arm is derived based on the iterative Newton–Euler formalism in terms of screw theory. The iterative method allows us to compute the local dynamic model of each joint and therefore to apply localized nonlinear controllers.

The authors show also the importance of the Hamiltonian in control theory. Instead of using the Lagrangian formulation of electromechanical or robotics systems, our work is focused on robot dynamics by its Hamiltonian. However, using a new recursive Newton–Euler algorithm, we generate the local Hamiltonians and the derivative of the moments at each joint of the robot manipulator. Thus, we can apply localized controllers at each joint.

References

1. Selig, J. (1992). *Introductory robotics*. Hertfordshire, UK: Prentice Hall International.
2. Lanczos, C. (1970). *The variational principles of mechanics* (4th ed., p. 92). New York: Dover Publications Inc.
3. Udwadia, F. E., & Kalaba, R. E. (2002). On the foundations of analytical dynamics. *International Journal of Non-linear Mechanics*, *37*(6), 1079–1090.
4. Woodhouse, N. M. J. (1987). *Introduction to analytical dynamics*. Oxford: Clarendon Press.
5. Abou El Dahab, E. T. (2000). A formulation of Hamiltonian mechanics using geometric algebra. *Advances in Applied Clifford Algebras*, *10*(2), 217–223.
6. Pappas, R. (1993). A formulation of Hamiltonian mechanics using geometric calculus In F. Brackx et al. (Eds.), *Clifford Algebras and Their Applications in Mathematical Physics* (pp. 251–258). Kluwer Academic Publishers.
7. Hestenes, D. (1986). *New foundations for classical mechanics*. Dordrecht: D. Reidel.
8. Hestenes, D., & Sobczyk, G. (1984). *Clifford algebra to geometric calculus: a unified language for mathematics and physics*. Dordrecht: D. Reidel.
9. Dress, A., & Havel, T. (1993). Distance geometry and geometric algebra. *Foundations of Physics*, *23*(10), 1357–1374.
10. Doran, C., Lasenby, A. (2005). *Geometric algebra for Physicists*. Cambridge University Press
11. Cederwall, M., & Salomon, P. (2009). *An introduction to analytical mechanics* (4th ed.). Göteborg: Chalmers University of Technology.
12. Craig, J. (2006). *Robotica*. Pearson.

Part III
Robot Control

Chapter 12
Control of Robot Manipulators

"the scientific study of control and communication in the animal and the machine." …Cybernetics includes the study of feedback, black boxes and derived concepts such as communication and control in living organisms, machines and organizations including self-organization [1948].

"Information is information, not energy or matter."

Norbert Wiener (11.26.1894–4.18.1964)

In this chapter, we present the localized control policy which is possible due to the computing of the local dynamic model at each robot joint computed using the recursive Newton–Euler algorithm. We compute the local Hamiltonians at each joint and derive their localized controllers as well. In the experimental part, we compare the performance of PD, Bang–Bang, and sliding mode controllers.

12.1 Robot Controllers Using Screw Theory

For constructing the law of control, it is necessary to compute the kinetic energy of the system. The kinetic energy can be written, in the screw theory framework, as a combination of the momentum co-screw and the screw

$$E_k = \frac{1}{2}(j \cdot w + p \cdot \mathbf{S}) = \frac{1}{2}\mathcal{P}(\mathbf{S}) = \frac{1}{2}N\mathbf{S}(\mathbf{S}) = \frac{1}{2}\mathbf{S}^T N\mathbf{S}, \tag{12.1}$$

where $\mathbf{S}$ stands for the twist $\boldsymbol{s}$ written now as a 6×1 vector $\mathbf{S} = [\omega, v]^T$. Now, to ensure that the robot reaches the desired position, it is necessary to consider the following theorems.

Theorem 12.1.1 *A robotic system (11.122)–(11.140) will reach a desired position applying the following localized law of control:*

© Springer Nature Switzerland AG 2020

E. Bayro-Corrochano, *Geometric Algebra Applications Vol. II*,
https://doi.org/10.1007/978-3-030-34978-3_12

$$\tau_i = K_{p_i}\widetilde{\theta}_i + K_{vi}\dot{\widetilde{\theta}}_i - \sum_{j=i}^{n} G_j \cdot L_i \;\; here \;\; i = 1, 2, \ldots, n, \tag{12.2}$$

where $\widetilde{\theta}_i = \theta_{di} - \theta_i$, θ_{di} *is the desired position in each joint,* θ_i *is the measured position of the robot in each joint, and the control gains for the position and velocity are* $K_{p_i}, K_{vi} \in \mathbb{R}^+$.

Proof To prove the convergence of the error, it is necessary to implement the Lyapunov theory, where the following Lyapunov function is proposed for a single-link robot:

$$V(\widetilde{\theta}) = E_k + \frac{1}{2}K_p\widetilde{\theta}^2. \tag{12.3}$$

Differentiating the Lyapunov function with respect to the time

$$\begin{aligned}\dot{\mathbf{S}}(\widetilde{\theta}) &= \frac{1}{2}\frac{d}{dt}(\mathbf{S}N\mathbf{S}) + \frac{1}{2}K_p\frac{d}{dt}\left(\widetilde{\theta}_i^2\right), \\ &= \frac{1}{2}\left(\frac{d}{dt}\left(\mathbf{S}^T\right)N\mathbf{S} + \mathbf{S}^T\frac{d}{dt}(N)\,\mathbf{S} + \mathbf{S}^T N\frac{d}{dt}(\mathbf{S})\right) + \frac{1}{2}K_p\frac{d}{dt}\left(\widetilde{\theta}^2\right),\end{aligned} \tag{12.4}$$

according to Eq. (11.78), the derivative $\dot{N}\mathbf{S} = \frac{d}{dt}(N)\,\mathbf{S} = \{\mathbf{S}, N\mathbf{S}\}$ and considering $\dot{\widetilde{\theta}} = -\dot{\theta}$

$$\begin{aligned}\dot{\mathbf{S}}(\widetilde{\theta}) &= \frac{1}{2}\left(\dot{\mathbf{S}}^T N\mathbf{S} + \mathbf{S}^T\dot{N}\mathbf{S} + \mathbf{S}^T N\dot{\mathbf{S}}\right) - K_p\widetilde{\theta}\dot{\theta}, \\ &= \dot{\mathbf{S}}^T N\mathbf{S} + \frac{1}{2}\mathbf{S}^T\{\mathbf{S}, N\mathbf{S}\} - K_p\widetilde{\theta}\dot{\theta}, \\ &= \dot{\mathbf{S}}^T N\mathbf{S} + \frac{1}{2}\cancel{\left(\mathbf{S}^T N[\mathbf{S}, \mathbf{S}]\right)} - K_p\widetilde{\theta}\dot{\theta} = \dot{\mathbf{S}}^T N\mathbf{S} - K_p\widetilde{\theta}\dot{\theta}.\end{aligned} \tag{12.5}$$

The equation of motion of a single rigid body is given by

$$\mathcal{W} = \mathrm{N}\dot{\mathbf{S}} + \{\mathbf{S}, \mathrm{N}\mathbf{S}\}. \tag{12.6}$$

Taking (12.6) into (12.46)

$$\dot{\mathbf{S}}(\widetilde{\theta}) = \mathbf{S}^T\mathcal{W} - \mathbf{S}^T\{\mathbf{S}, N\mathbf{S}\} - K_p\widetilde{\theta}\dot{\theta}, \tag{12.7}$$

$$= \mathbf{S}^T\mathcal{W} - \cancel{\mathbf{S}^T N[\mathbf{S}, \mathbf{S}]} - K_p\widetilde{\theta}\dot{\theta} = \mathbf{S}^T\mathcal{W} - K_p\widetilde{\theta}\dot{\theta}. \tag{12.8}$$

For a single-link $\mathcal{W} = \mathcal{T} + \mathcal{R} + G$

$$\dot{\mathbf{S}}(\widetilde{\theta}) = \mathbf{S}^T(\mathcal{T} + \mathcal{R} + G) - K_p\widetilde{\theta}\dot{\theta}. \tag{12.9}$$

Considering $\mathbf{S} = L\dot{\theta}$, $\tau = L \cdot \mathcal{T}$ and $\mathbf{S}^T\mathcal{R} = 0$

$$\dot{\mathbf{S}}(\widetilde{\theta}) = (\tau + G \cdot L)\,\dot{\theta} - K_p \widetilde{\theta}\dot{\theta}. \tag{12.10}$$

Applying the law of control $\tau = K_p\widetilde{\theta} + K_v\dot{\widetilde{\theta}} - G^T L$

$$\dot{\mathbf{S}}(\widetilde{\theta}) = K_V \dot{\widetilde{\theta}}\dot{\theta} = -K_v\dot{\widetilde{\theta}}^2. \tag{12.11}$$

Here, (12.11) is not a globally asymptotically stable system. But using the LaSalle's theorem, it is possible to conclude that $\widetilde{\theta} = 0$ if and only if $\dot{\widetilde{\theta}} = 0$. Therefore, the robotic system with the law of control proposed is globally asymptotically stable [1]. In addition, using (11.113), it is possible to extend the previous proof for a robot with n DOF, where the control for any serial robot can be written as (11.122).

Theorem 12.1.2 *A robotic system (11.122)–(11.140) will reach the home position applying the following localized law of control:*

$$\tau_i = K_i sign(\mathcal{S}_i) + K_{\mathcal{S}i}\mathcal{S}_i + \widetilde{\theta}_i - \sum_{j=i}^{n} \mathbf{S}_j^T \{\mathbf{S}_i, N_j L_i\} - G_j \cdot L_i, \tag{12.12}$$

where $\{\mathbf{S}_i, N_j L_i\} = N_j\,[\mathbf{S}_i, L_i]$ *and in each joint,* $\mathcal{S}_i = -\sum_{j=i}^{n} L_i^T N_j \mathbf{S}_j + \widetilde{\theta}_i$ *is the sliding surface and* $K_i, K_{\mathcal{S}i} \in \mathbb{R}^+$.

Proof In sliding mode control [2], it is necessary to prove the convergence of the surface. Therefore, the following Lyapunov function is proposed:

$$V(\mathcal{S}_i) = \frac{1}{2}\mathcal{S}_i^2 = \frac{1}{2}\mathcal{S}_i\mathcal{S}_i. \tag{12.13}$$

Differentiating the Lyapunov function with respect to the time, one gets

$$\dot{\mathbf{S}}(\mathcal{S}_i) = \mathcal{S}_i\dot{\mathcal{S}}_i = -\mathcal{S}_i \frac{d}{dt}\left(\sum_{j=i}^{n} L_i^T N_j \mathbf{S}_j + \widetilde{\theta}_i\right), \tag{12.14}$$

$$= -\mathcal{S}_i \sum_{j=i}^{n} \left(\dot{L}_i^T N_j \mathbf{S}_j + L_i^T \dot{N}_j \mathbf{S}_j + L_i^T N_j \dot{\mathbf{S}}_j + \widetilde{\theta}_i\right),$$

$$= -\mathcal{S}_i \sum_{j=i}^{n} \left(\mathbf{S}_j^T N_j\,[\mathbf{S}_i, L_i] + L_i^T \left\{\mathbf{S}_j, N_j \mathbf{S}_j\right\} + L_i^T N_j \dot{\mathbf{S}}_j + \widetilde{\theta}_i\right),$$

$$= -\mathcal{S}_i \sum_{j=i}^{n} \left(\mathbf{S}_j^T N_j\,[\mathbf{S}_i, L_i] + \mathbf{S}_j^T N_j \left[L_i, \mathbf{S}_j\right] + L_i^T N_j \dot{\mathbf{S}}_j + \widetilde{\theta}_i\right). \tag{12.15}$$

Taking (11.122) into (12.15):

$$\dot{\mathbf{S}}(\mathcal{S}_i) = -\mathcal{S}_i \left(\tau_i + \dot{\theta}_i + \sum_{j=i}^{n} \mathbf{S}_j^T N_j \left[\mathbf{S}_i, L_i \right] + G_j \cdot L_i \right). \tag{12.16}$$

Applying the law of control of Eq. (12.12) in Eq. (12.16)

$$\tau_i = K_i \text{sign}(\mathcal{S}_i) + K_{\mathcal{S}i} \mathcal{S}_i + \widetilde{\theta}_i - \sum_{j=i}^{n} \mathbf{S}_j^T \{ \mathbf{S}_i, N_j L_i \} - G_j \cdot L_i, \tag{12.17}$$

$$\dot{\mathbf{S}}(\mathcal{S}_i) = -K_i |\mathcal{S}_i| - K_{\mathcal{S}i} \mathcal{S}_i \cdot \mathcal{S}_i. \tag{12.18}$$

Hence, the sliding surface will converge to zero. Then, it is possible to see that $\mathcal{S}_i = 0 \rightarrow \theta_i = 0$. Therefore, the robotic system with the law of control proposed will reach the home position.

12.2 Experimental Analysis of Controllers Using Screw Theory

Consider the two-link manipulator with revolute joints illustrated in Fig. 12.1. In addition, the manipulator link lengths are l_1 and l_2, the link masses are m_1 and m_2, where these masses are concentrated at the end of each link.

The dynamic of the proposed robot is computed using (11.122). Hence, the dynamic of each joint can be written as the following:

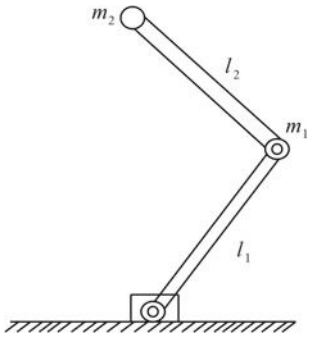


Fig. 12.1 2-DOF robot manipulator

$$\begin{aligned}\tau_1 &= \dot{\mathbf{S}}_1^T N_1 L_1 + \mathbf{S}_1^T N_1 [L_1, \mathbf{S}_1] - G_1 \cdot L_1 + \dot{\mathbf{S}}_2^T N_2 L_1, \\ &\quad + \mathbf{S}_2^T N_2 [L_1, \mathbf{S}_2] - G_2 \cdot L_1, \\ \tau_2 &= \dot{\mathbf{S}}_2^T N_2 L_2 + \mathbf{S}_2^T N_2 [L_2, \mathbf{S}_2] - G_2 \cdot L_2.\end{aligned} \tag{12.19}$$

Considering $\mathbf{S}_1 = L_1\dot{\theta}_1$ and $\mathbf{S}_2 = L_1\dot{\theta}_1 + L_2\dot{\theta}_2$, the accelerations are constructed by

$$\dot{\mathbf{S}}_1 = \frac{d}{dt}\left(L_1\dot{\theta}_1\right) = L_1\ddot{\theta}, \tag{12.20}$$

$$\begin{aligned}\dot{\mathbf{S}}_2 &= \frac{d}{dt}\left(L_1\dot{\theta}_1 + L_2\dot{\theta}_2\right) = L_1\ddot{\theta}_1 + L_2\ddot{\theta}_2 + ad(\mathbf{S}_1)L_2\dot{\theta}_2, \\ &= L_1\ddot{\theta}_1 + L_2\ddot{\theta}_2 + [L_1, L_2]\,\dot{\theta}_1\dot{\theta}_2.\end{aligned} \tag{12.21}$$

Taking the previous accelerations $(\dot{V}_1, \dot{V}_2)$ and velocities (V_1, V_2) into (12.50) and (12.51)

$$\begin{aligned}\tau_1 &= L_1^T N_1 L_1 \ddot{\theta}_1 + \cancel{L_1^T N_1 [L_1, L_1]\dot{\theta}_1^2} + \\ &\quad + L_1^T N_2 \left(L_1\ddot{\theta}_1 + L_2\ddot{\theta}_2 + [L_1, L_2]\,\dot{\theta}_1\dot{\theta}_2\right) + \\ &\quad + \left(L_1\dot{\theta}_1 + L_2\dot{\theta}_2\right)^T N_2 \left[L_1, \left(\cancel{L_1\dot{\theta}_1} + L_2\dot{\theta}_2\right)\right] - G_1^T L_1 - G_2^T L_1,\end{aligned} \tag{12.22}$$

$$\begin{aligned}\tau_2 &= L_2^T N_2 \left(L_1\ddot{\theta}_1 + L_2\ddot{\theta}_2 + \cancel{[L_1, L_2]\dot{\theta}_1\dot{\theta}_2}\right) + \\ &\quad + \left(L_1\dot{\theta}_1 + \cancel{L_2\dot{\theta}_2}\right)^T N_2 \left[L_2, L_1\dot{\theta}_1 + \cancel{L_2\dot{\theta}_2}\right] - G_2^T L_2.\end{aligned} \tag{12.23}$$

Then, the dynamic of the robot can be written as follows:

$$\begin{aligned}\tau_1 &= L_1^T N_1 L_1 \ddot{\theta}_1 + L_1^T N_2 L_1 \ddot{\theta}_1 + L_1^T N_2 L_2 \ddot{\theta}_2 + 2L_1^T N_2 [L_1, L_2]\,\dot{\theta}_1\dot{\theta}_2 + \\ &\quad + L_2^T N_2 [L_1, L_2]\,\dot{\theta}_2^2 - G_1^T L_1 - G_2^T L_1,\end{aligned} \tag{12.24}$$

$$\tau_2 = L_1^T N_2 L_2 \ddot{\theta}_1 + L_2^T N_2 L_2 \ddot{\theta}_2 - L_1^T N_2 [L_1, L_2]\,\dot{\theta}_1^2 - G_2^T L_2. \tag{12.25}$$

Now, the closed-loop system can be represented in state variables. Suppose $x_1, x_2 \in \mathbb{R}^2$, where $x_1 = \left(\widetilde{\theta}_1, \widetilde{\theta}_2\right)^T$ and $x_2 = \left(\dot{\widetilde{\theta}}_1, \dot{\widetilde{\theta}}_2\right)^T$.
PD controller: The closed-loop system using Theorem 12.1.1 can be illustrated by

$$\begin{pmatrix} \dot{x_1} \\ \dot{x_2} \end{pmatrix} = \begin{pmatrix} x_2 \\ -\begin{pmatrix} L_1^T N_1 L_1 + L_1^T N_2 L_1 & L_1^T N_2 L_2 \\ L_1^T N_2 L_2 & L_2^T N_2 L_2 \end{pmatrix}^{-1} \left(\left(\begin{pmatrix} K_{p1} & 0 \\ 0 & K_{p2} \end{pmatrix} x_1 \right. \right. \\ \left. \left. + \begin{pmatrix} K_{v1} & 0 \\ 0 & K_{v2} \end{pmatrix} x_2 - B(\dot{\theta}_1, \dot{\theta}_2) \right) \end{pmatrix}, \tag{12.26}$$

Table 12.1 Parameters of the robot

Parameter	Value	Unit
m_1	0.25	kg
m_2	0.25	kg
l_1	1.2	m
l_2	1.2	m

where

$$B(\dot{\theta}_1, \dot{\theta}_2) = \begin{pmatrix} 2L_1^T N_2 [L_1, L_2] \dot{\theta}_1 \dot{\theta}_2 + L_2^T N_2 [L_1, L_2] \dot{\theta}_2^2 \\ -L_1^T N_2 [L_1, L_2] \dot{\theta}_1^2 \end{pmatrix}. \tag{12.27}$$

To simulate the results of the example, we use the parameters in Table 12.1 with $K_{p1}, K_{p2} = 150$ and $K_{v1}, K_{v2} = 50$. The simulations were conducted using the Euler integration method, with a step size of 0.001 s. The initial conditions were selected as $x_1(0) = x_2(0) = 0$ and the desired values as $\theta_{d1} = 20°, \theta_{d2} = 30°$. The results of the example are presented in Fig. 12.2. We didn't apply the PD controller for tracking, because in its formalism the dynamic of tracking signal is not considered, thus PD tracking could not be robust, in contrast this can be done using sliding modes as shown next.

Sliding Mode Controller: Suppose $x_1, x_2 \in \mathbb{R}^2$, where $x_1 = (\theta_1, \theta_2)^T$ and $x_2 = \left(\dot{\theta}_1, \dot{\theta}_2\right)^T$, then the closed-loop system using Theorem 12.1.2 can be formulated as follows:

$$\begin{pmatrix} \dot{x_1} \\ \dot{x_2} \end{pmatrix} = \begin{pmatrix} x_2 \\ -\begin{pmatrix} L_1^T N_1 L_1 + L_1^T N_2 L_1 & L_1^T N_2 L_2 \\ L_1^T N_2 L_2 & L_2^T N_2 L_2 \end{pmatrix}^{-1} \left(\begin{pmatrix} K_1 \text{sign}(\mathcal{S}_1) + K_{\mathcal{S}1} \\ K_2 \text{sign}(\mathcal{S}_2) + K_{\mathcal{S}2} \end{pmatrix} \right. \\ \left. -B(x_2) + x_2) \end{pmatrix}, \tag{12.28}$$

where

$$\begin{aligned} \mathcal{S}_1 &= L_1^T N_1 L_1 \dot{\theta}_1 + L_1^T N_2 L_1 \dot{\theta}_1 + L_1^T N_2 L_2 \dot{\theta}_2 + \theta_1, \\ \mathcal{S}_2 &= L_1^T N_1 L_2 \dot{\theta}_1 + L_2^T N_2 L_2 \dot{\theta}_2 + \theta_2, \\ B(x_2) &= \begin{pmatrix} 2L_1^T N_2 [L_1, L_2] \dot{\theta}_1 \dot{\theta}_2 + L_2^T N_2 [L_1, L_2] \dot{\theta}_2^2 \\ L_2^T N_2 [L_1, L_2] \dot{\theta}_1 \dot{\theta}_2 \end{pmatrix}. \end{aligned} \tag{12.29}$$

To simulate the results of the example, the parameters of Table 12.1 with $K_1, K_2 = 1$ and $K_{\mathcal{S}1}, K_{\mathcal{S}2} = 50$ are used. The simulations were conducted using the Euler integration method, with a step size of 0.001 s. The initial conditions were selected as $x_1(0) = [0°, 0°]^T$ and $x_2(0) = [0, 0]^T$. Using the sliding mode controller, we carry out a tracking experiment. Note that any perturbation belonging to the control space will be rejected by the SM controller. Moreover, it is necessary to take care with the adjustable gains, because a small variable cannot easily converge to the position for

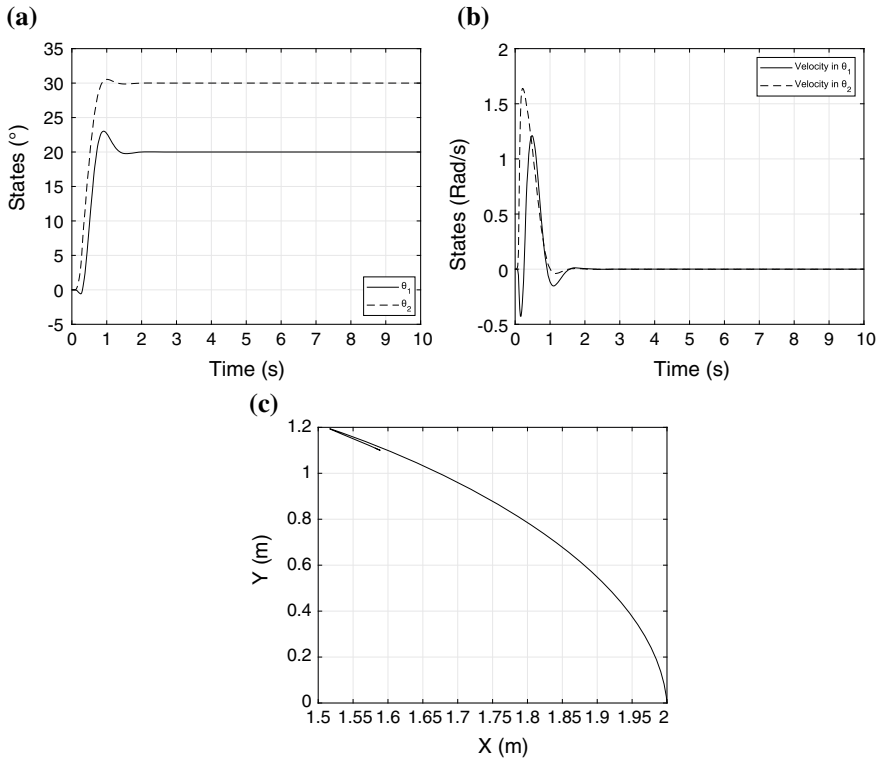


Fig. 12.2 PD Controller: **a** States: angles x_1 and angular velocities x_2. **b** Robot motion. **c** Robot final position $x = 1.58$ mt., $y = 1.1$ mt.

the case of different robots. Therefore, it is important to adjust the gain depending upon the application.

Figure 12.3a–b presents the evolution of the angles and angular velocities of the two joints during the tracking. This experiment shows that our sliding mode controller tracks very well the nonlinear curve. Figure 12.3c–d shows how good the sliding mode controller manages the 2-DOF robot arm to follow the circle. Note that any matched perturbation which even though belongs to the control space will be rejected anyway by the SM controller.

12.3 Robot Controllers Using Hamiltonians

In this section, we study the application of localized controllers derived using the local Hamiltonian equation (11.332) and local derivative momenta Eq. (11.335) for robot systems. The used decentralized controllers are the Bang–Bang, a PD, and sliding mode controllers [2].

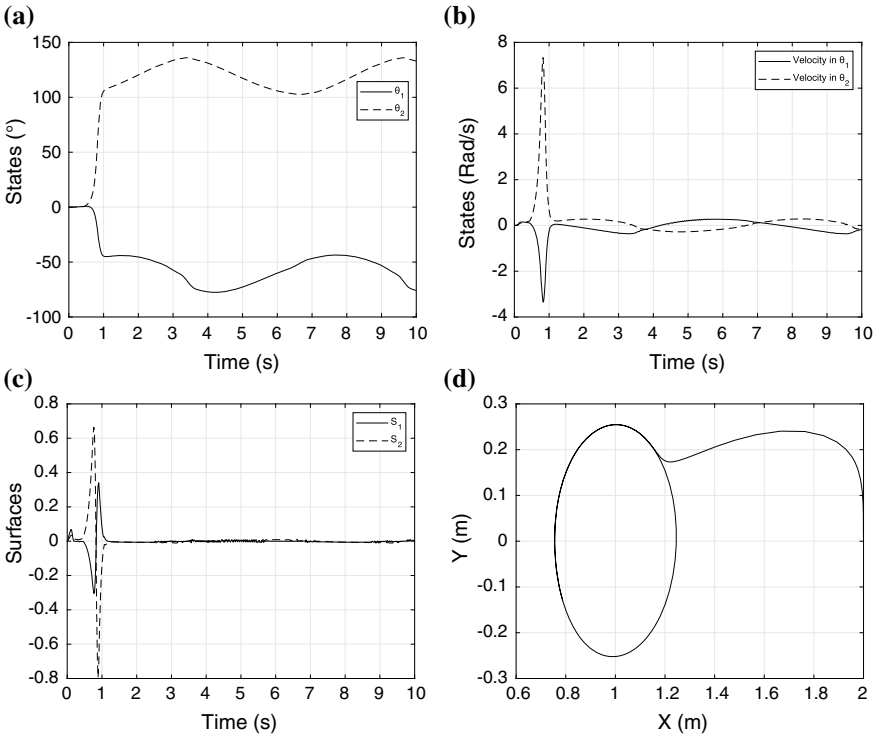


Fig. 12.3 Sliding mode controller: **a** states: angles x_1; **b** angular velocities x_2; **c** sliding mode surfaces; **d** robot motion

12.3.1 Localized Hamiltonian Bang–Bang Control

Using the Newton–Euler recursive algorithm of Sect. 11.10, we can compute inward for each joint the following local Hamiltonians using generalized coordinates (Fig. 12.4)

$$H_2 = \frac{1}{2} m_2 l_2^2 \dot{\theta}_2^2 - m_2 g l_2 \cos(\theta_2), \tag{12.30}$$

$$H_1 = \frac{1}{2}(m_1 + m_2) l_1^2 \dot{\theta}_1^2 + m_2 l_1 l_2 \dot{\theta}_1 \dot{\theta}_2 \cos(\theta_1 - \theta_2) - (m_1 + m2) g l_1 \cos(\theta_1).$$

Since $H_1(0) = 0$ and $H_2(0) = 0$, Eq. (12.30) becomes

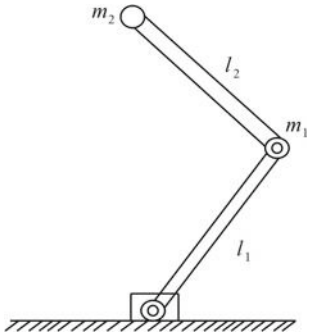


Fig. 12.4 2-DOF planar robot

$$H_2(\theta_2) = \frac{1}{2} m_2 l_2^2 \dot{\theta}_2^2 + m_2 g l_2 (1 - \cos(\theta_2)), \tag{12.31}$$

$$H_1(\theta_1) = \frac{1}{2}(m_1 + m_2) l_1^2 \dot{\theta}_1^2 + m_2 l_1 l_2 \dot{\theta}_1 \dot{\theta}_2 \cos(\theta_1 - \theta_2) + \\ + (m_1 + m2) g l_1 (1 - \cos(\theta_1)).$$

The Bang–Bang control is applied in a localized manner at each joint using the local Hamiltonians similar as in the state Eq. (12.34). These control signals $u_i = \pm 1, i = 1, 2$ switch between the scalar functions of H_1 and H_2.

$$H_2(\theta_2) = \frac{1}{2} m_2 l_2^2 \dot{\theta}_2^2 + m_2 g l_2 (1 - \cos(\theta_2)) - b_2 \theta_2 u_2, \tag{12.32}$$

$$H_1(\theta_1) = \frac{1}{2}(m_1 + m_2) l_1^2 \dot{\theta}_1^2 + m_2 l_1 l_2 \dot{\theta}_1 \dot{\theta}_2 b \cos(\theta_1 - \theta_2) + \\ + (m_1 + m2) g l_1 (1 - \cos(\theta_1)) - b_1 \theta_1 u_1. \tag{12.33}$$

12.4 PD and Sliding Mode Controllers for Localized Hamiltonian Control

Using the Newton–Euler recursive algorithm, we can compute the tensors of the Euler–Lagrange state equation with control signals as follows:

$$M \begin{bmatrix} \ddot{\theta}_1 \\ \\ \ddot{\theta}_2 \end{bmatrix} + C \left(\begin{bmatrix} \dot{\theta}_1 \\ \\ \dot{\theta}_2 \end{bmatrix}, \begin{bmatrix} \theta_1 \\ \\ \theta_2 \end{bmatrix} \right) + G(\theta_1, \theta_2) = \begin{bmatrix} b_1 u_1 \\ \\ b_2 u_2 \end{bmatrix}, \tag{12.34}$$

where

$$M = \begin{bmatrix} (m_1 + m_2)l_1^2 & m_2 l_1 l_2 \cos(\theta_1 - \theta_2) \\ m_2 l_1 l_2 \cos(\theta_1 - \theta_2) & m_2 l_2 \end{bmatrix},$$

$$C\left(\begin{bmatrix} \ddot{\theta}_2 \\ \ddot{\theta}_1 \end{bmatrix}, \begin{bmatrix} \dot{\theta}_2 \\ \dot{\theta}_1 \end{bmatrix}\right) = \begin{bmatrix} 0 & m_2 l_1 l_2 \dot{\theta}_2 \sin(\theta_1 - \theta_2) \\ -m_2 l_1 l_2 \dot{\theta}_1 \sin(\theta_1 - \theta_2) & 0 \end{bmatrix},$$

$$G\left(\begin{bmatrix} \theta_2 \\ \theta_1 \end{bmatrix}\right) = \begin{bmatrix} (m_1 + m_2) g l_1 \sin\theta_1 \\ m_2 g l_2 \sin\theta_2 \end{bmatrix}.$$

Before constructing the law of control, it is necessary to compute Hamilton's equations in the screw theory framework. The Hamiltonian on the phase space can be computed using Eq. (11.331) as follows:

$$\mathcal{H}(\theta, p, t) = \sum_{j=1}^{n} \frac{1}{2} \mathbf{P}_j^T N_j^{-1} \mathbf{P}_j + \widetilde{g}^T \widetilde{c}_j, \tag{12.35}$$

$$\frac{\partial \mathcal{H}(\theta, p, t)}{\partial p_i} = \frac{1}{\mathbf{S}_i^T N_i \mathbf{S}_i} \left(p_i - \sum_{j=i+1}^{n} \mathbf{S}_i^T \mathbf{P}_j - \sum_{k=1}^{i-1} \mathbf{S}_i^T \mathbf{P}_{i,k} \right), \tag{12.36}$$

$$\frac{\partial \mathcal{H}(\theta, p, t)}{\partial \theta_i} = \sum_{j=i}^{n} \mathbf{P}_j^T \left[\mathbf{S}_i, N_i^{-1} \mathbf{P}_i \right] - G_j^T \mathbf{S}_i, \tag{12.37}$$

where $\mathbf{S}_i$ is the twist $\boldsymbol{s}_i$ written as a 6×1 vector and the momentum $\mathbf{P}_i$ is a 6×1 vector as well.

According to Eq. (11.336), the derivative of the momenta at i-joint is

$$\dot{p}_i = \tau_i + \sum_{j=i}^{n} G_j^T \mathbf{S}_i - \mathbf{P}_j^T \left[\mathbf{S}_i, N_i^{-1} \mathbf{P}_i \right]. \tag{12.38}$$

Now, to ensure that the robot reaches the desired position, it is necessary to consider the following theorems.

Theorem 12.4.1 *A serial robotic system will track a desired smooth function applying the following localized law of control:*

$$\tau_i = K_{p_i} \widetilde{\theta}_i + K_{v_i} \widetilde{p}_i + \dot{p}_{d_i} - \sum_{j=i}^{n} \mathbf{P}_j^T \left[N_i^{-1} \mathbf{P}_i, \mathbf{S}_i \right] - G_j^T \mathbf{S}_i, \tag{12.39}$$

where $\widetilde{p}_i = p_{di} - p_i$ *is the error between the desired momentum and the measured momentum,* $\widetilde{\theta}_i = \theta_{di} - \theta_i$ *is the error between the desired joint position and the measured joint position and* $K_{p_i}, K_{vi} \in \mathbb{R}^+$.

Proof The proof of this theorem can be easily solved. Taking the law of control in Eq. (12.39) into Eq. (12.38) gives

$$K_{p_i}\widetilde{\theta}_i + K_{vi}\widetilde{p}_i + \dot{\widetilde{p}}_i = 0. \tag{12.40}$$

The previous equation is linear and the closed-loop system is globally asymptotically stable, if and only if, $K_{p_i}, K_{vi} \in \mathbb{R}^+$ [1]. This law of control is similar to the traditional PD-CTC (computed-torque control with a proportional–derivative action). Based upon the foregoing, the closed-loop system can be represented by

$$\dot{\widetilde{p}}_i = -K_{p_i}\widetilde{\theta}_i - K_{vi}\widetilde{p}_i. \tag{12.41}$$

Theorem 12.4.2 *A serial robotic system will reach a desired position applying the following localized law of control:*

$$\tau_i = K_i sign(\mathcal{S}_i) + K_{\mathcal{S}i}\mathcal{S}_i + \frac{1}{\mathbf{S}_i^T N_i \mathbf{S}_i}\left(\widetilde{p}_i + \sum_{j=i+1}^{n}\mathbf{S}_i^T\mathbf{P}_j + \sum_{k=1}^{i-1}\mathbf{S}_i^T\mathbf{P}_{i,k}\right) -$$
$$- \sum_{j=i}^{n}\mathbf{P}_j^T\left[N_i^{-1}\mathbf{P}_i, \mathbf{S}_i\right] - G_j^T\mathbf{S}_i, \tag{12.42}$$

where $\mathcal{S}_i = \widetilde{p}_i + \widetilde{\theta}_i$ *is the sliding surface,* $\widetilde{p}_i = p_{di} - p_i$ *is the error between the desired momentum and the measured momentum,* $\widetilde{\theta}_i = \theta_{di} - \theta_i$ *is the error between the desired joint position and the measured joint position and* $K_i, K_{\mathcal{S}i} \in \mathbb{R}^+$.

Proof In sliding mode control, it is necessary to prove the convergence of the surface [2]. Therefore, it is proposed by the following Lyapunov function:

$$V(\mathcal{S}_i) = \frac{1}{2}\mathcal{S}_i^2 = \frac{1}{2}\mathcal{S}_i\mathcal{S}_i. \tag{12.43}$$

Differentiating the Lyapunov function with respect to the time, we get

$$\dot{\mathbf{S}}(\mathcal{S}_i) = \mathcal{S}_i\dot{\mathcal{S}}_i = \mathcal{S}_i\frac{d}{dt}\left(\widetilde{p}_i + \widetilde{\theta}_i\right). \tag{12.44}$$

Taking $\dot{\widetilde{\theta}}_i = -\dot{\theta}_i$, Eq. (11.122), Eq. (12.36), and Eq. (12.38)

$$\dot{\mathbf{S}}(\mathcal{S}_i) = \mathcal{S}_i\Big(- \tau_i - \sum_{j=i}^{n} \mathbf{P}_j^T \left[N_i^{-1}\mathbf{P}_i, \mathbf{S}_i \right] - G_j^T \mathbf{S}_i - \\ - \frac{1}{\mathbf{S}_i^T N_i \mathbf{S}_i}(p_i - \sum_{j=i+1}^{n} \mathbf{S}_i^T \mathbf{P}_j - \sum_{k=1}^{i-1} \mathbf{S}_i^T \mathbf{P}_{i,k})\Big). \tag{12.45}$$

Applying the law of control in Eq. (12.42)

$$\dot{\mathbf{S}}(\mathcal{S}_i) = -K_i|\mathcal{S}_i| - K_{\mathcal{S}i}\mathcal{S}_i^2. \tag{12.46}$$

Hence, the sliding surface will converge to zero. Then, it is possible to see that $\mathcal{S}_i = 0 \rightarrow \widetilde{\theta}_i = 0$. Therefore, the robotic system with the law of control proposed will reach the desired position [?]. Based upon the foregoing, the closed-loop system can be represented by

$$\dot{\widetilde{p}}_i = -K_i \text{sign}(\mathcal{S}_i) - K_{\mathcal{S}i}\mathcal{S}_i - \frac{1}{\mathbf{S}_i^T N_i \mathbf{S}_i}\left(\widetilde{p}_i + \sum_{j=i+1}^{n} \mathbf{S}_i^T \mathbf{P}_j + \sum_{k=1}^{i-1} \mathbf{S}_i^T \mathbf{P}_{i,k} \right). \tag{12.47}$$

12.5 Experimental Analysis of Controllers Using the Robot Hamiltonians

In this section, we study the control of a planar 2-DOF robot arm formulated from the Hamiltonian of the robotic system. The tested controllers are the Bang–Bang controller, the PD controller, and the sliding mode controller.

12.5.1 Bang–Bang Control

Applying in Eqs. (12.32)–(12.33) the maximal principle of Pontryagin and setting the angle references $\theta_{1_{ref}} = 20°$ and $\theta_{2_{ref}} = 30°$ and $u_i = \pm 1, i = 1, 2$ for the switching in the Hamiltonians so that the Hamiltonians reach the zero value, we obtain the following equations:

$$H_{11} = \frac{1}{2}(m_1 + m_2)l_1^2\dot{\theta}_1^2 + m_2 l_1 l_2 \dot{\theta}_1 \dot{\theta}_2 \cos\left((\theta_1 - \theta_{1_{ref}}) - (\theta_2 - \theta_{2_{ref}})\right) + \\ + (m_1 + m2)gl_1(1 - \cos(\theta_1 - \theta_{1_{ref}})) - b_1(\theta_1 - \theta_{1_{ref}}),$$

$$H_{12} = \frac{1}{2}(m_1 + m_2)l_1^2\dot{\theta}_1^2 + m_2 l_1 l_2 \dot{\theta}_1 \dot{\theta}_2 \cos\left((\theta_1 - \theta_{1_{ref}}) - (\theta_2 - \theta_{2_{ref}})\right) + \\ + (m_1 + m2)gl_1(1 - \cos(\theta_1 - \theta_{1_{ref}})) + b_1(\theta_1 - \theta_{1_{ref}}),$$

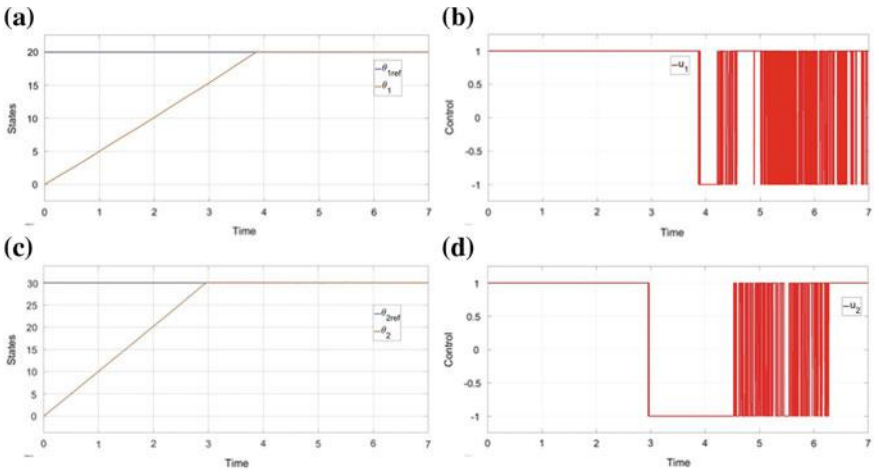


Fig. 12.5 Hamiltonian Bang–Bang controller: **a** state θ_1; **b** control u_1; **c** state θ_2; **d** control u_2;

where $u_1 =$ if $H_{12} > H_{11}$ then -1 else 1 and

$$H_{21} = \frac{1}{2} m_2 l_2^2 \dot{\theta}_2^2 + m_2 g l_2 (1 - \cos(\theta_2 - \theta_{2_{ref}})) - b_2(\theta_2 - \theta_{2_{ref}}), \quad (12.48)$$

$$H_{22} = \frac{1}{2} m_2 l_2^2 \dot{\theta}_2^2 + m_2 g l_2 (1 - \cos(\theta_2 - \theta_{2_{ref}})) + b_2(\theta_2 - \theta_{2_{ref}}), \quad (12.49)$$

where $u_2 =$ if $H_{22} > H_{21}$ then -1 else 1. The control signals $u_i = \pm 1, i = 1, 2$ switch between the Hamiltonian H_1 and H_2 surfaces. Figure 12.5 shows the control signals and the error figures to reach a reference point. Note that even though the controllers reach the references, they keep switching very rapidly between ± 1.

12.5.2 PD and Sliding Mode Control

Suppose the two-link manipulator with revolute joints shown in Fig. 12.4. The manipulator link lengths are l_1 and l_2, the link masses are m_1 and m_2, where these masses are concentrated at the end of each link.

Hamilton's equations of the proposed robot are computed using Eqs. (12.36), (12.37), and (12.38). Hence, these equations, in the proposed robot, are written as

$$\frac{\partial \mathcal{H}(\theta, p, t)}{\partial p_1} = \frac{1}{\mathbf{S}_1^T N_1 \mathbf{S}_1} \left(p_1 - \mathbf{S}_1^T \mathbf{P}_2\right), \quad (12.50)$$

$$\frac{\partial \mathcal{H}(\theta, p, t)}{\partial p_2} = \frac{1}{\mathbf{S}_2^T N_2 \mathbf{S}_2} \left(p_2 - \mathbf{S}_2^T \mathbf{P}_{2,1}\right), \quad (12.51)$$

$$\frac{\partial \mathcal{H}(\theta, p, t)}{\partial \theta_1} = \mathbf{P}_1^T \left[\mathbf{S}_1, N_1^{-1}\mathbf{P}_1\right] + \mathbf{P}_2^T \left[\mathbf{S}_1, N_1^{-1}\mathbf{P}_1\right] - G_1^T \mathbf{S}_1 - G_2^T \mathbf{S}_1, \quad (12.52)$$

$$\frac{\partial \mathcal{H}(\theta, p, t)}{\partial \theta_2} = \mathbf{P}_2^T \left[\mathbf{S}_2, N_2^{-1}\mathbf{P}_2\right] - G_2^T \mathbf{S}_2, \quad (12.53)$$

$$\dot{p}_1 = \tau_1 + G_1^T \mathbf{S}_1 + G_2^T \mathbf{S}_1 - \mathbf{P}_1^T \left[\mathbf{S}_1, N_1^{-1}\mathbf{P}_1\right] - \mathbf{P}_2^T \left[\mathbf{S}_1, N_1^{-1}\mathbf{P}_1\right], \quad (12.54)$$

$$\dot{p}_2 = \tau_2 + G_2^T \mathbf{S}_2 - \mathbf{P}_2^T \left[\mathbf{S}_2, N_2^{-1}\mathbf{P}_2\right]. \quad (12.55)$$

Now, the closed-loop system of the example can be represented using Eq. (12.47). Then, the closed-loop system using Theorem 12.1.1 can be illustrated by

$$\dot{\tilde{p}}_1 = -K_1 \text{sign}(\mathcal{S}_1) - K_{\mathcal{S}1}\mathcal{S}_1 - \frac{1}{\mathbf{S}_1^T N_1 \mathbf{S}_1}\left(\tilde{p}_1 + \mathbf{S}_1^T \mathbf{P}_2\right), \quad (12.56)$$

$$\dot{\tilde{p}}_2 = -K_2 \text{sign}(\mathcal{S}_2) - K_{\mathcal{S}2}\mathcal{S}_2 - \frac{1}{\mathbf{S}_2^T N_2 \mathbf{S}_2}\left(\tilde{p}_2 + \mathbf{S}_2^T \mathbf{P}_{21}\right), \quad (12.57)$$

where

$$\begin{aligned} \mathcal{S}_1 &= \tilde{\theta}_1 + \tilde{p}_1, \\ \mathcal{S}_2 &= \tilde{\theta}_2 + \tilde{p}_2. \end{aligned} \quad (12.58)$$

To simulate the results of the example, we use the parameters in Table 12.2.

The simulations were conducted using the Euler integration method, with a step size of 0.001 s. The initial conditions were selected as $\theta_1(0) = \theta_2(0) = 0°$, $p_1(0) = p_2(0) = 0\ k_g \frac{m}{s}$ and the desired values as $\theta_{d1} = 175°$, $\theta_{d2} = 45°$. Hence, the results of the example are presented in Fig. 12.6.

On the other hand, using Theorem 12.1.2, the closed-loop system of the example can be represented using Eq. (12.41). Hence, the closed-loop system is illustrated as follows:

$$\dot{\tilde{p}}_1 = -K_{p1}\tilde{\theta}_1 - K_{v1}\tilde{p}_1, \quad (12.59)$$

$$\dot{\tilde{p}}_2 = -K_{p2}\tilde{\theta}_2 - K_{v2}\tilde{p}_2. \quad (12.60)$$

To simulate the results of the example, we use the parameters in Table 12.3.

Table 12.2 Parameters of the robot

Parameter	Value	Unit
$m_1,\ m_2$	0.15	kg
$l_1,\ l_2$	1.5	m
K_1, K_2	50	
$K_{\mathcal{S}1}, K_{\mathcal{S}2}$	1	

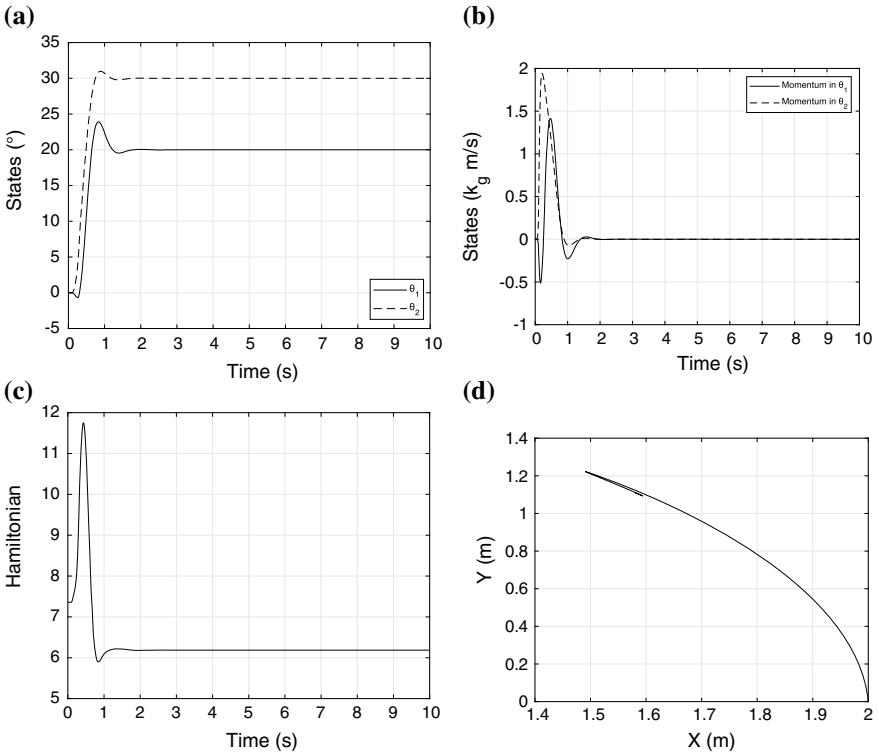


Fig. 12.6 PD controller: state variables: **a** angles; **b** momenta; **c** Hamiltonian; **d** robot motion

Table 12.3 Parameters of the robot

Parameter	Value	Unit
$m_1,\ m_2$	0.15	kg
$l_1,\ l_2$	1.5	m
K_{p1}, K_{p2}	1	
K_{v1}, K_{v2}	1	

The simulations were conducted using the Euler integration method, with a step size of 0.001 s. The initial conditions were selected as $\theta_1(0) = -45°$, $\theta_2(0) = 0°$, $p_1(0) = p_2(0) = 0\ k_g \frac{m}{s}$ and the desired values as $\theta_{d1} = \theta_{d2} = 2t°$. The results of the example using a SM controller are presented in Fig. 12.7. Note that any perturbation belonging to the control space will be rejected by the SM controller.

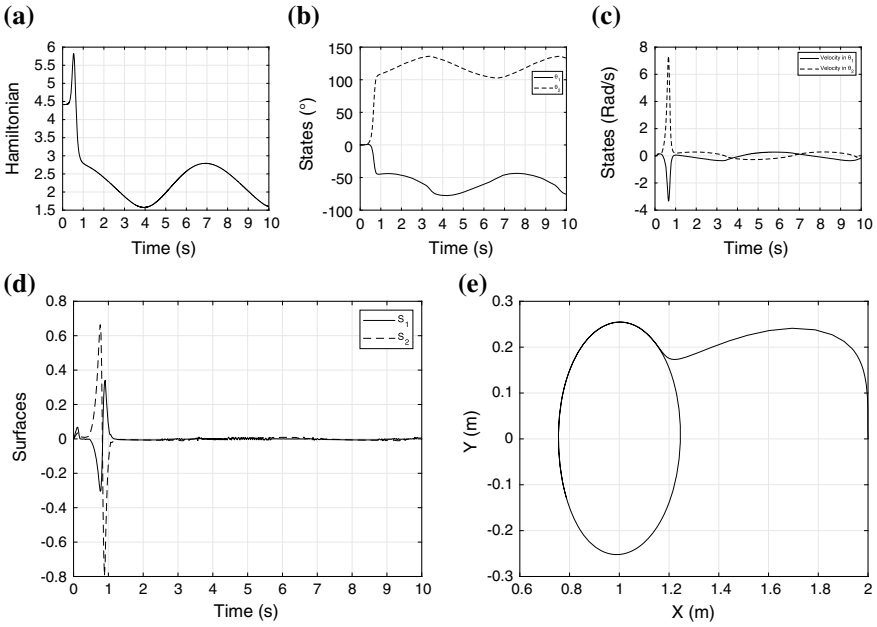


Fig. 12.7 Sliding mode controller: **a** Hamiltonian; state variables; **b** angles; **c** momenta; **c** surfaces; **d** tracking robot motion

12.6 Conclusion

This work presents the dynamic model and the nonlinear control for robot manipulators using the geometric algebra framework specifically utilizing the motor algebra $G^{+}_{3,0,1}$. The dynamic model of a robot arm is derived based on the recursive Newton–Euler formalism in terms of screw theory. The recursive method allows us to compute the local dynamic model of each joint and therefore to apply localized nonlinear controllers. A PD and sliding mode controllers in terms of screws were proposed. The stability of the controllers is also proved. The controllers have the advantage of being easily programmed and they can be used for any serial robot, and the DOF are not relevant for the calculation. In the experimental analysis, we compare a PD and a sliding mode controller. The experimental analysis shows that our control law is stable and tracks a nonlinear trajectory.

In the second part, the authors show the importance of the Hamiltonian in control theory. Instead of using the Lagrangian formulation of electromechanical or robotics systems, our work is focused on robot dynamics by its Hamiltonian. However, using a new recursive Newton–Euler algorithm, we generate the local Hamiltonians and the derivative of the moments at each joint of the robot manipulator. Thus, we can

apply localized controllers at each joint. So, we work in phase space and study the performance of three control methods: the Bang–Bang control, the PD control, and sliding modes. In the experimental part, we compare and discuss the efficiency of these controllers.

References

1. Khalil, H. K. (2002). *Nonlinear systems*. East Lansing: Pearson.
2. Utkin, V., Guldner, J., & Shi, J. (1999). *Sliding mode control in electromechanic systems*. Taylor and Francis.

Chapter 13
Robot Neurocontrol

Deep learning, also called deep structural learning or hierarchical learning, is a set of algorithms in machine learning that attempt to model high-level abstractions of space–time data by using model architectures composed of multiple nonlinear transformations and recurrent connections.

13.1 Introduction

Biological creatures are able to perform complex tasks, due to the capacity of the brain to store information and to adapt its neuro connections as necessary, and this is known as synaptic plasticity [1]. The neuroplasticity was investigated and later used in Artificial Neural Networks (ANN), where these ANN were called the third generation of neural networks [2]. The main advantage of the third generation of neural networks, or Spiking Neural Networks (SNN), is the ability to mimic the biological behavior, where this characteristic can be used in a variety of applications.

The SNN has the property to be a kind of adaptive controller, where this ability can be used to control a robot [3]. On the other hand, it is proved that the quaternion neural networks provide more benefits than the conventional neural networks [4]. Therefore, in this work, we extend the spiking neural networks using quaternions, that is, we formulate the Quaternion Spiking Neural Network (QSNN).

13.2 Quaternion Spiking Neural Networks

The design of spike neural networks is inspired by the way of biological neuron works in living organisms, which concentrate all energy from neighboring neurons to the dendrites and that is redirected to the soma, where it is accumulated. If the excitation signal is strong enough to overcome the neuronal threshold, it causes a

© Springer Nature Switzerland AG 2020

E. Bayro-Corrochano, *Geometric Algebra Applications Vol. II*,
https://doi.org/10.1007/978-3-030-34978-3_13

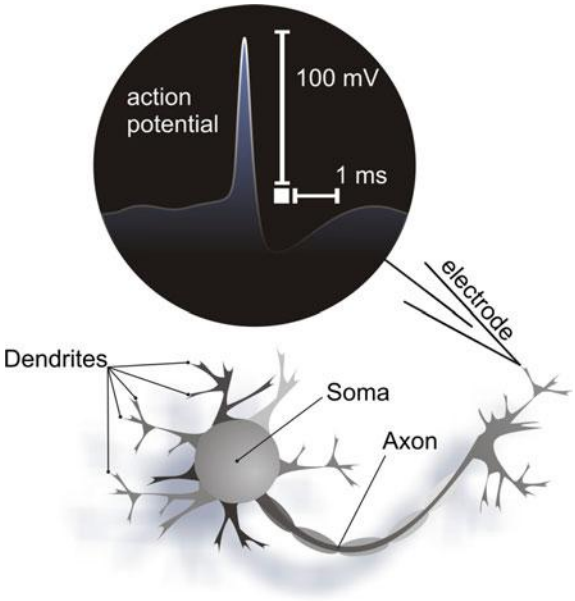


Fig. 13.1 Dendrite, soma, and axon can be clearly distinguished in the figure. The circle shows an example of a neuronal action potential. The action potential is a relatively short voltage pulse of 1–2 ms duration and an amplitude of about 100 mV

potential boost action, which will travel through the axon to the postsynaptic neurons closing the neuro-activity cycle [5].

The passage of electrical potential from the axon of one neuron to the dendrite of another neuron is called synapse. It is common to refer to the sending neuron as the presynaptic neuron and the receptor neuron as postsynaptic. A single neuron in the cerebral cortex of vertebrates often is connected to more than 10^4 postsynaptic neurons [6]. Neural signals consist of short electrical impulses and can be observed by placing a thin electrode near the soma or axon of a neuron, Fig. 13.1. The pulses have an amplitude of about 100 mV and usually a duration of 1–2 ms. The pulse shape (amplitude and period) is always the same in all neurons of the cerebral cortex so the action potential carries no information [7]. Rather, the way the brain handles the information is in the number and the time at which these pulses occur [8].

13.2.1 Neuronal Model

At present, there are a variety of mathematical models that emulate the behavior of biological neural networks, which extend from models with enormous biological plausibility as proposed by Hodgkin–Huxley [5] until relatively simple models such as integration and shot having the minimum number required to meet the characteristics which describe the behavior of neuron properties but with the advantage of having a very low computational cost [9].

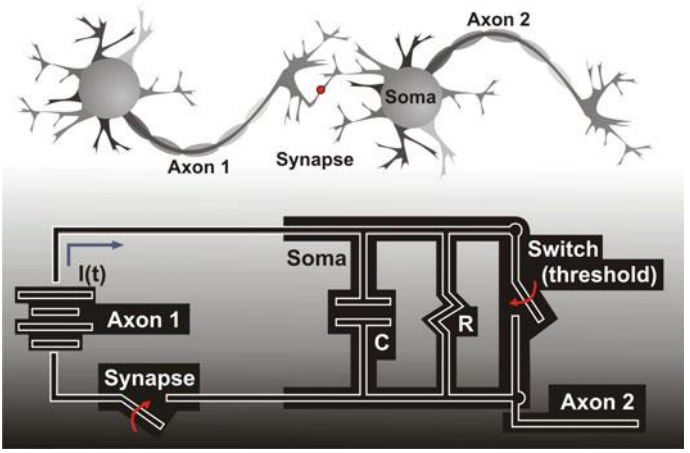


Fig. 13.2 The LIF model as a capacitive resistive circuit. When the threshold value is exceeded, the capacitor enters for a moment in a breakdown voltage, and this way the circuit emits a pulse of output. If there does not exist a presynaptic pulse, the capacitor will eventually discharge through the Jule effect caused by the resistance

The type of neuron that is used in this paper is a neural model of integration and fire (LIF). The LIF model is represented by a parallel RC circuit shown in Fig. 13.2, where the energy stored in the capacitor is dissipated not only when the threshold is exceeded, but also by the resistive component of the circuit.

The differential equation describing this system is given by

$$V(t) = V_r e^{-\frac{1}{RC}(t-t_0)} + \frac{1}{C}\int_{t_0}^{t} I(\tau) e^{-\frac{1}{RC}(t-\tau)} t\tau, \tag{13.1}$$

where $V(t)$ is the capacitor voltage, V_R is the voltage value in rest of the capacitor, C is the value of capacitance in micro Farads, $I(t)$ is the value of the current source, and R is the resistance value in milliohms.

The PIF y LIF models are included as examples of integrate and fire models and they have been extensively implemented in several research papers [7], because of their simple implementation and low computational cost. The threshold models represent dynamics with a high abstraction level, and they are based on the summation of all the contributions of the presynaptic neurons to the membrane's potential, so that in case of exceeding the previously given threshold value, the neuron fires. The threshold models are, in fact, divided into two sub-models: Perfect Integrate and Fire (PIF) and Leaky Integrate and Fire (LIF).

The first-order and ordinary differential equation describing the behavior of the integrate and fire neurons does not completely specify the behavior of the third-generation artificial neural networks, since it only embraces the dynamic starting from the resting potential conditions until the instant of time just before the neuron

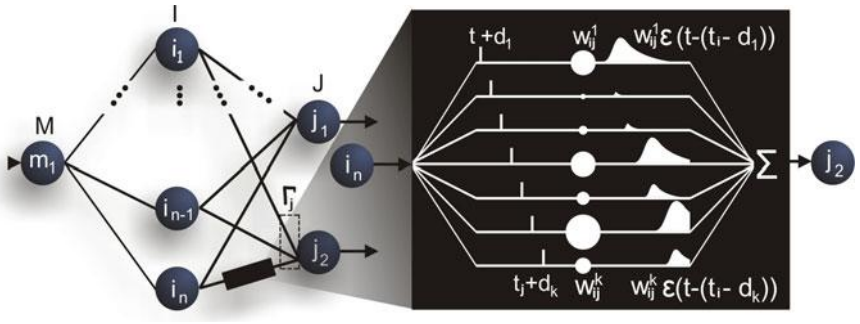


Fig. 13.3 Model of a simple neural network. The postsynaptic neuron layer J receives the input set Γ_j (the signals of the j-th presynaptic neuron). The connection between the interconnection of the neuron i-th to j-th neuron shows that each connection between two neurons has m delayed synaptic terminals. Each synaptic terminal k is the k-th delay (d_k) which is associated with a synaptic weight k-th between i-th neuron to neuron j-th ($W_J I^K$) [10]. White circles dictate the magnitude of synaptic weights

fires (the precise instant of time where the receptor neuron emits the output pulse), so that the conditions of firing must be posed. Equation (13.2) expresses this fact by saying that $t^{(f)}$ is the precise moment where the given threshold θ starts to be exceeded and that at this instant of time the membrane's potential is an increasing function.

$$t^{(f)} : u(t^{(f)}) = \theta \quad \text{and} \quad \left.\frac{du(t)}{dt}\right|_{t=t^{(f)}} > 0. \tag{13.2}$$

13.2.2 Neuronal Architecture

The spiking neural network is a feedforward architecture with original presynaptic signal (see Fig. 13.3), as described by Ruf and NatschlNager [10], inspired by the work of Haberly (1985) where he wrote that existing in the pyriform cortex of the brain of a rat is a wide range of delays (greater than 20 ms) between two adjacent brain areas.

The dynamic behavior described by the input potential to the neuron in a time interval is

$$x_j = \sum_{i \in D_j} w_{ij} \epsilon(t - t_i), \tag{13.3}$$

where $\epsilon(t)$ describes the response function of spike type and w_{ij} is the weight that describes the value of the synapse. The response function type spike $\epsilon(t)$ is given by

$$\alpha(t) = \frac{t}{T} e^{1-\frac{t}{T}} \Theta, \tag{13.4}$$

where T is the decay constant of the membrane potential, which determines the rise and fall time of postsynaptic potential and Θ is the Heaviside function.

The number of connections that have a neuron in this architecture is the number of neurons of the preceding layer, and this is shown in Fig. 13.3 where the neuron j-th has n presynaptic connections and each of these connections is composed of k delays (d_k). Each of these delays is defined as the difference between the time that the firing of the presynaptic neuron and the time that postsynaptic potential begins to grow occurs [11]. So the equation that describes each of these delays is

$$y_i^k = \alpha(t - t_i - d_k), \tag{13.5}$$

where time t_i is the time when the presynaptic neuron i issues a pulse and d_k is the delay associated with the synaptic terminal k.

Considering Eq. (13.3), which describes the behavior which occurs when adding the synapses of all neurons belonging to the previous layer with the neuron in the next layer, and the equation that considers multiple attached delays that exist in each synaptic connection Eq. (13.4), one can formulate the final equation which describes the full dynamic between a presynaptic layer and postsynaptic neurons X_j as follows:

$$X_j(t) = \sum_{i \in D_j} \sum_{k=1}^{m} w_{ij}^k y_i^k(t). \tag{13.6}$$

13.3 Training Algorithm

This section describes the implemented neural training, which is based on the delta rule proposed by Widrow and Hoff 1960 at Stanford University [12] and although it is a method designed to make a linear adaptation, it can perfectly work with nonlinear functions [13].

Similar to other training algorithms, adjusting network weights is carried out through iterations to associate the input to the neuron with a desired output. It starts from the rule proposed by adjusting weights Widrow–Hoff

$$\Delta w_i^{WH} = \lambda y_i (y_d^o - y^o), \tag{13.7}$$

where $\lambda \in R$ refers to the rate of learning, y_i is the i-th neural connection without being scaled by the corresponding associated weight to this connection (note: for now delays are not considering), y_d^o and y^o refer to the desired and actual output neurons, respectively. Originally this learning rule was designed to neural networks with linear activation functions and real values, but, in our case, we will use to change the actual value of the vectors which are nonlinear functions in which the neural model is evaluated.

To adjust Eq. (13.7) for be able to work with spiking neural networks and its architecture with delayed entries is first defined the k-th delay of the i-th input neuron (y_i^k) as

$$y_i^k(t) = \sum \alpha_k(t - t_i - d_k), \tag{13.8}$$

where $\alpha(t)$ refers to the α function defined in Eq. (13.4), and this way you can get the right moment where occurs the pulse as a continuous function. Similarly, in this way, it is defined as the desired output neuronal and the current output neuronal equations

$$y_d^{ko}(t) = \sum \alpha_k(t - t_d^o - d_k), \tag{13.9}$$

$$y^{ko}(t) = \sum \alpha_k(t - t^o - d_k). \tag{13.10}$$

To end the adequacy of the law Widrow–Hoff in the spike networks architecture with multiple delays, one substitutes in the integer of the variables of Eq. (13.7), Eqs. (13.8), (13.9), and (13.10), and in this way, we are doing a mapping with an integer in the interval time in which the neural model is defined, so you get a scalar with the necessary information to get the desired output

$$\Delta w_i^k = \lambda \int y_i^k(t)(y_d^o(t) - y^o(t))dt. \tag{13.11}$$

Defining the error as the difference between the desired output signal and the actual output signal, Eq. (13.11) becomes

$$\Delta w_i^k = \lambda \int y_i^k(t)E(t)dt. \tag{13.12}$$

In Fig. 13.4, the adjustment process of the weights is shown, based on the Widrow–Hoff rule extended to spiking neural networks until up a step before integration.

13.3.1 Rotation with Quaternions

The rotations in $\mathbb{R}^3$ are expressed by the subgroup $SO(3)$. Wherein quaternions, the rotor operator is illustrated as

$$Q = \cos\frac{\theta}{2} + \sin\frac{\theta}{2}\boldsymbol{v}, \tag{13.13}$$

where θ is the rotation angle and $\boldsymbol{v}$ is an arbitrary axis of rotation. On the other hand, the conjugate of the rotor is

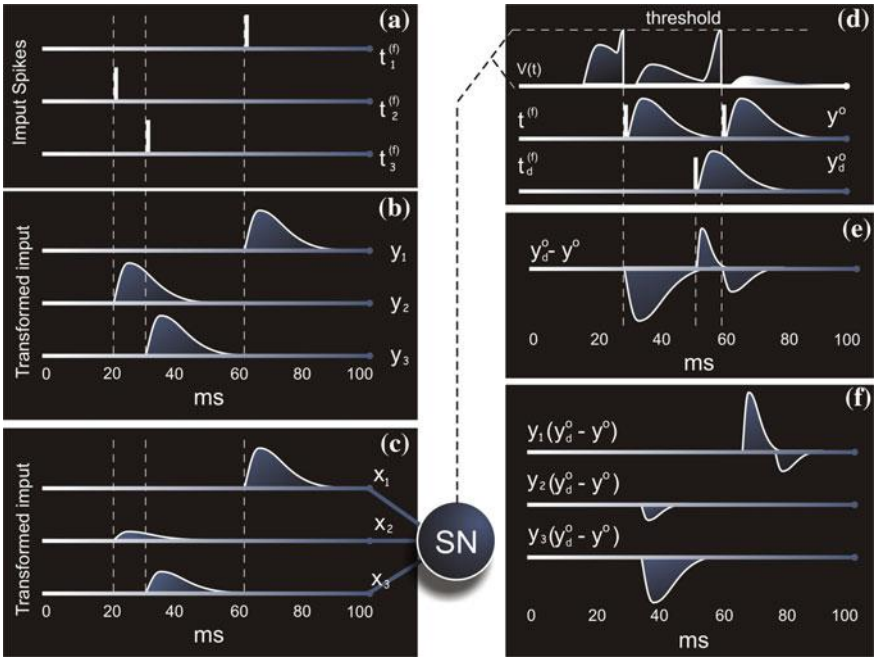


Fig. 13.4 Training algorithm: **a** wherein the set of input pulses is shown. In this case, it was not considered the multiple delays by synapse because the adjustment process of the weight is the same for any time in which a pulse occurs. **b** The transformation of each pulse shown with α-kernel. **c** Inputs y_i are multiplied by their respective weights w_i, the superposition of these signals are evaluated in the neural model. **d** First line shows the dynamic membrane potential, and each time is exceeded the threshold a pulse is outputted (y^o) which is compared with the desired pulse (y_d^o) generating an error. **e** Multiplies and integrates the error ($y_d^o - y^o$) per each entry (y_i) thereby obtaining each one of the weights' modify (Δw_i)

$$\widetilde{Q} = \cos\frac{\theta}{2} - \sin\frac{\theta}{2}\boldsymbol{v}. \tag{13.14}$$

Then, the rotation of a given point $P(P_0\boldsymbol{i}, P_1\boldsymbol{j}, P_2\boldsymbol{k})$ around an arbitrary axis ($\boldsymbol{v}$) with angle θ can be expressed as

$$P' = Q \bigotimes P \bigotimes \widetilde{Q}, \tag{13.15}$$

where $\bigotimes$ represents the quaternion multiplication.

13.3.2 Quaternion Spike Neural Network

To convert the Widrow–Hoff rule into quaternions (see Fig. 13.5), the Widrow–Hoff rule is modified as

1. The input variable $In(t) \in \mathbb{R}^1$ is coded to $t_m \in \mathbb{R}^n$, being n the number of elements that has the vector in which the neural model will be evaluated.
2. The coded signal (t_m) is delayed with

$$t_m + d * D; \qquad D = 0, 1, 2, \ldots, p, \tag{13.16}$$

   where d is the delayed constant and D is the number of delays. With the totality of the delays the matrix of $p \times n$ is formed, we will call this matrix as R-matrix.
3. The R-matrix is expanded to quaternion with

$$\boldsymbol{i} = R + d; \ \boldsymbol{j} = \boldsymbol{i} + d; \ \boldsymbol{k} = \boldsymbol{j} + d. \tag{13.17}$$

4. To each delayed vector of these matrices are applied the kernel-transformation

$$\varepsilon = \frac{t - (t_m^q + d * D)}{\tau} e^{1 - \frac{t-(t_m^q + d*D)}{\tau}}. \tag{13.18}$$

5. The quaternionic matrix is transformed to get the input to the postsynaptic neuron as

$$V(t) = \sqrt{\overline{(I_q \bigotimes W_q)} \cdot (I_q \bigotimes W_q) I_r}. \tag{13.19}$$

6. $V(t)$ is evaluated in the neural model. So, if the membrane potential exceeds the threshold, the neuron will emit a spike in that instant and it will be the next t_m.
7. The response of the neuron enters in the next layer, if it corresponds to the output layer, then the output layer signal is compared to find an error signal. This error signal helps to adjust the weights as follows:

$$\Delta \boldsymbol{W}_i = \lambda \int (y_d^o(t) - y^o(t)) \bigotimes \overline{\boldsymbol{y}}_i(t) dt. \tag{13.20}$$

13.4 Control of a Simulated Nonlinear System

Inspired by the fact that spike neural networks under the proposed quaternion input are capable of reproducing nonlinear functions, arise the idea of using this architecture such that its neurons process nonlinear functions. In this way, we could formulate a control law for nonlinear systems, being designed and adapted in real time.

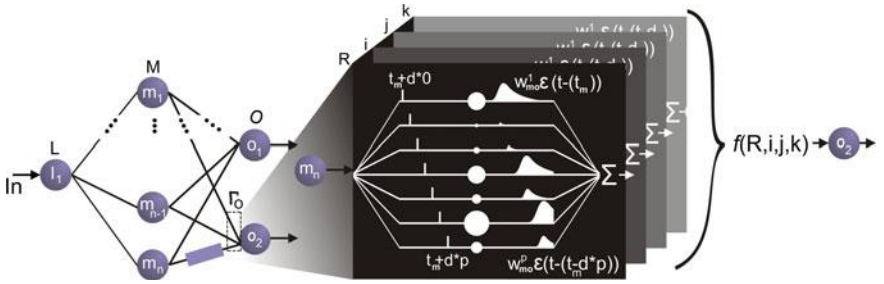


Fig. 13.5 Extension of the Widrow–Hoff rule training to a quaternion rule for the spike neural network

Given a nonlinear system proposed using arbitrarily difference equations as follows:

$$X(k) = (40.93X(k-1) + 1.91\cos(0.79X(k-2)) - 3.93\sin(0.97X(k-3)) + 0.95(u(k) + u(k-1))e^{0.0037u(k-2)^2}))4.31, \tag{13.21}$$

where $X(k)$, $X(k-1)$, $X(k-2)$, and $X(k-3)$ are the system response at the current sampling time, and one, two, and three delayed sampling periods, respectively, and in the same way for the control law $u(k)$, $u(k-1)$, and $u(k-2)$. As can be seen in the proposed plant, is an unstable system of type SISO, even the control signal is modified internally with an exponential function.

The proposed closed-loop control uses the response of the system and its derivative, as well as the SP (Fig. 13.6); the neural network was designed as a feedforward architecture, with three layers and one neuron per layer, and each neuron receives as input signal the system response, its derivative, and the desired reference plus the output of the neuron from the previous layer; the three input signals to the first neuron are sufficient so that the neural network creates a function in four dimensions, and this function (evaluated by the neural network in those three input parameters) is injected as a control law to the system and the created error of the difference between the desired output minus the system response is fed back to the neural network to adjust the weights and to modify the four-dimensional function, in this way, closing the control cycle.

In Fig. 13.7, the response of the system through the first 15 training epochs is shown, and it can be observed that as the training grows the system response converges to the desired reference.

In Fig. 13.8, the evolution of the error between the desired reference and the system response over time is shown, where each epoch is equal to 36,000 iterations and the reason is that the desired reference cycle is completed (remember, one training epoch in neural networks is when it is completely contemplated all the input vectors); therefore, Fig. 13.8 shows the response of the system in the first 15 epochs. In the same figure, an approach is made for the last 5 epochs, where it is seen that the

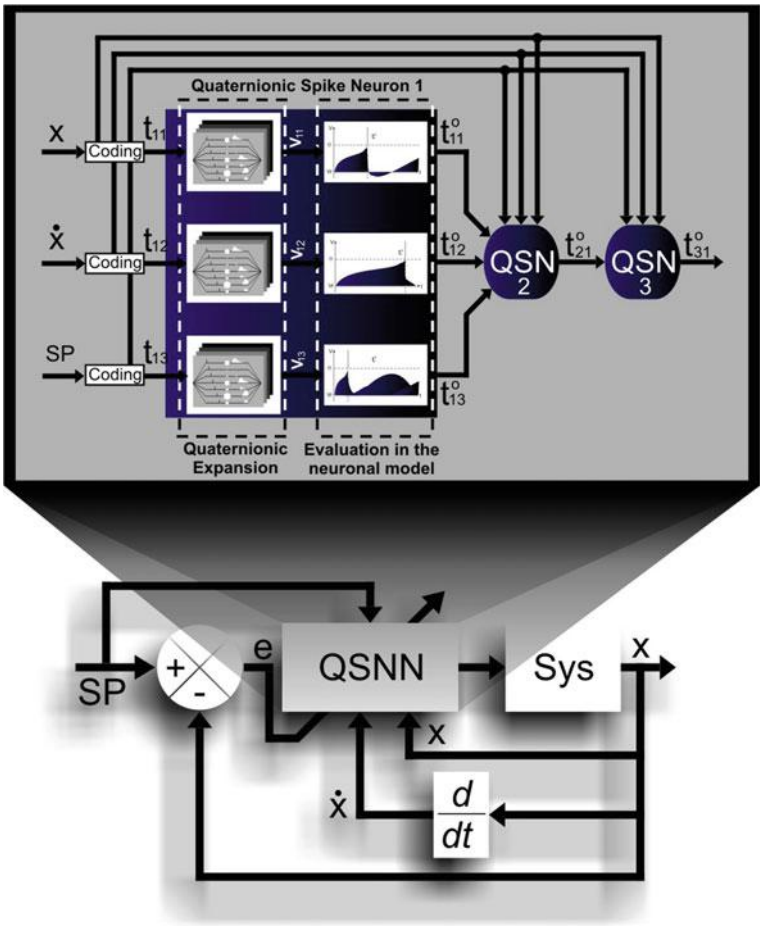


Fig. 13.6 Control scheme: the upper part shows the neural architecture used, describing what happens inside the neuron in the first layer

error does not converge to zero and stays oscillating in ± 0.45 units and this is equal to an oscillating error of 1.5%, which occurs because the plasticity of the neural network is not able to generate a function that completely satisfies the relationship between the input variables and the desired output. This is a small cost compared to the advantage of controlling a system as complex as Eq. (13.21) without the inconvenience of needing its mathematical model to design the law of control.

Figures 13.9 and 13.10 show the control law designed by the QSNN. In Fig. 13.9, a classical two-dimensional way is shown, which shows a comparison between two control laws at different training periods (epoch 1 and epoch 15), and the form of the control law in the first epoch (intermittent blue line) generates the system response observed in Fig. 13.7 cyan color line, while after 15 training epochs (black

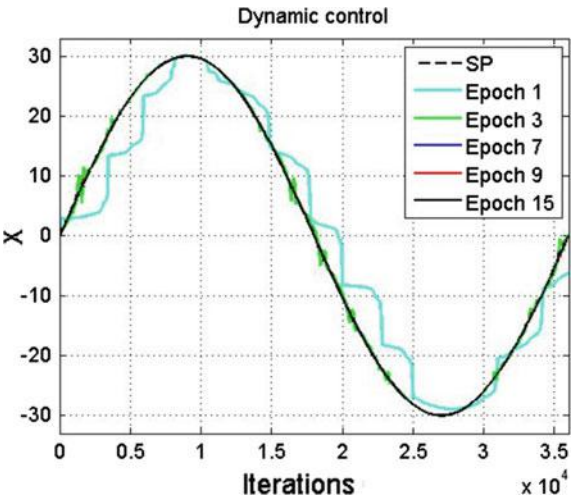


Fig. 13.7 The evolution of the system response at different epochs, it is possible to see how the response (variable x) converges to the desired signal while the training rise

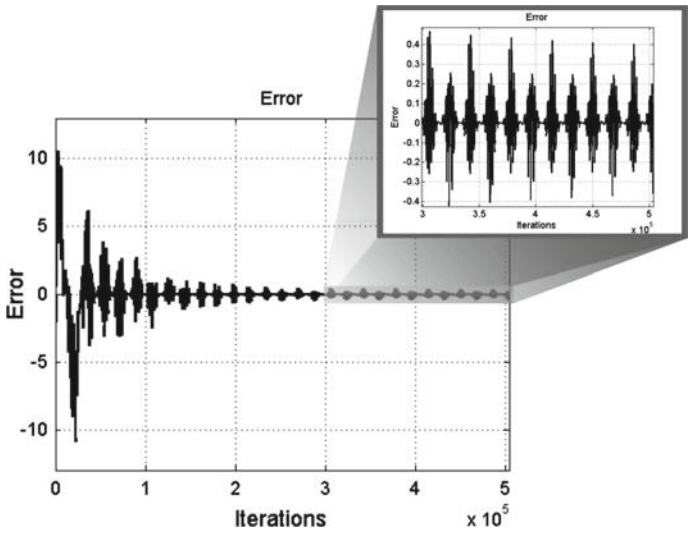


Fig. 13.8 The evolution of the error through the first 540 k iterations (15 epochs), in addition to a zoom in the last 5 epochs

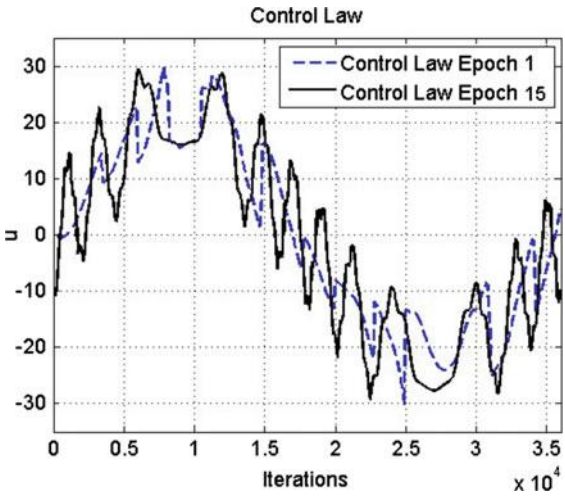


Fig. 13.9 The control law proposed by the quaternion spiking neural network at different epochs

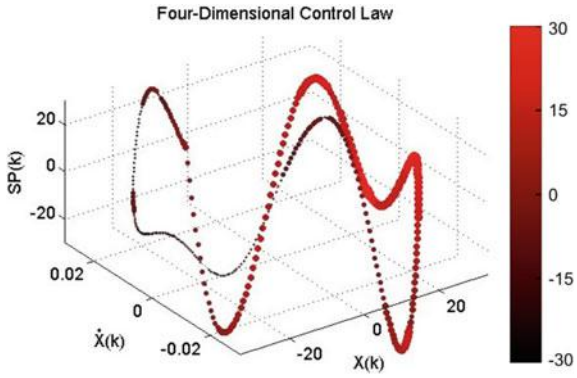


Fig. 13.10 The four-dimensional control law, with three Euclidean dimensions (system state, its derivative, and the desired reference) plus a fourth dimension in color form between $[-30, 30]$ (Black $= -30$, Red $= 30$)

line) the function proposed by the QSNN makes the system follow the desired signal better (oscillating error of 1.5). And now in Fig. 13.10, the four-dimensional function is shown, which satisfies the relationship between the three input variables to the network (system state, its derivative, and the desired reference) and the control law. Since it is not possible to visualize a complete four-dimensional scan as has been done in the X-OR problem section, it has been proposed to see to the fourth dimension as a color whose value varies between $[-30, 30]$ (Black $= -30$, Red $= 30$), and is only evaluated at the three-dimensional point entering to the neural network.

13.5 Control of a Real Nonlinear System

As we have seen, the quaternion spike neural networks are able to control nonlinear systems of relative order 2 or less, so the idea arises to apply them to the control of a humanoid robot arm (Mex One). To perform this task, it is proposed to obtain the direct and inverse kinematics with an innovative method based on quaternions, and subsequently the desired positions in each joint are used as desired references so that the quaternion spike neural network can carry each motor to a desired position.

13.5.1 Forward Kinematics

The forward kinematics model establishes the relationship between the position and orientation of the end-effector to the base [14]. Where in this work, the study is focused on the arm. So, to solve the forward kinematics model, it is necessary to settle the coordinates of each articulation as shown in Fig. 13.11.

Consequently, in accordance with the MexOne robot, the arm has six degrees of freedom. Therefore, there exist six quaternionic rotors as follows:

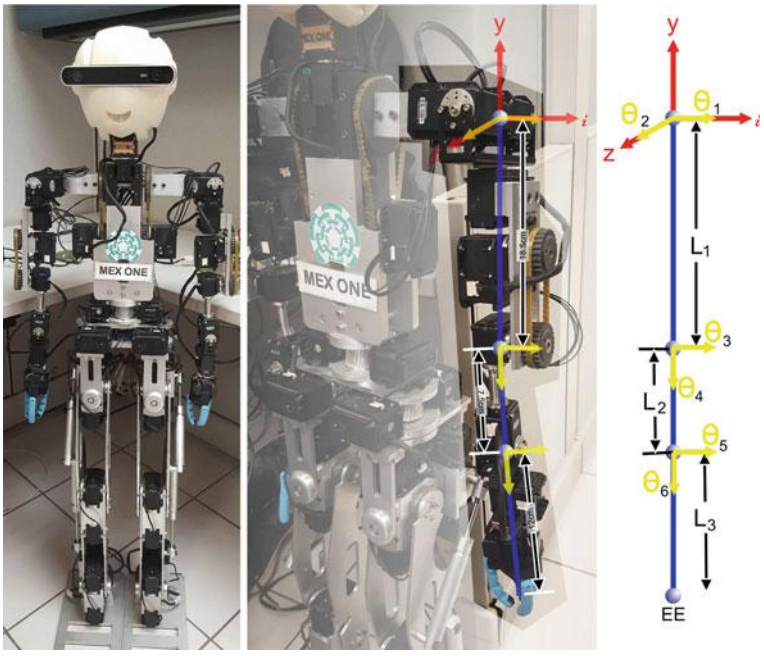


Fig. 13.11 Robot MexOne coordinates

$$Q_1 = \cos\frac{\theta_1}{2} + \sin\frac{\theta_1}{2}\boldsymbol{i}, \tag{13.22}$$

$$Q_2 = \cos\frac{\theta_2}{2} + \sin\frac{\theta_2}{2}\boldsymbol{k}, \tag{13.23}$$

$$Q_3 = \cos\frac{\theta_3}{2} + \sin\frac{\theta_3}{2}\boldsymbol{i}, \tag{13.24}$$

$$Q_4 = \cos\frac{\theta_4}{2} - \sin\frac{\theta_4}{2}\boldsymbol{j}, \tag{13.25}$$

$$Q_5 = \cos\frac{\theta_5}{2} + \sin\frac{\theta_5}{2}\boldsymbol{i}, \tag{13.26}$$

$$Q_6 = \cos\frac{\theta_6}{2} - \sin\frac{\theta_6}{2}\boldsymbol{j}. \tag{13.27}$$

So, the solution of the kinematic model can be written as

$$\begin{aligned} P_F = {} & Q_1 Q_2 L_1 \widetilde{Q}_2 \widetilde{Q}_1 + Q_1 Q_2 Q_3 Q_4 L_2 \widetilde{Q}_4 \widetilde{Q}_3 \widetilde{Q}_2 \widetilde{Q}_1 \\ & + Q_1 Q_2 Q_3 Q_4 Q_5 Q_6 L_3 \widetilde{Q}_6 \widetilde{Q}_5 \widetilde{Q}_4 \widetilde{Q}_3 \widetilde{Q}_2 \widetilde{Q}_1, \end{aligned} \tag{13.28}$$

where the elements of Eq. (13.28) can be written as

$$Q_1 Q_2 L_1 \widetilde{Q}_2 \widetilde{Q}_1 = \begin{bmatrix} L_1 s_1 & \boldsymbol{i} \\ -L_1 c_1 c_2 & \boldsymbol{j} \\ -L_1 s_1 c_2 & \boldsymbol{k} \end{bmatrix} \tag{13.29}$$

$$Q_1 Q_2 Q_3 Q_4 L_2 \widetilde{Q}_4 \widetilde{Q}_3 \widetilde{Q}_2 \widetilde{Q}_1 = \begin{bmatrix} L_2(s_2 c_3) & \boldsymbol{i} \\ L_2(s_1 s_3 - c_1 c_2 c_3) & \boldsymbol{j} \\ -L_2(c_1 s_3 + s_1 c_2 c_3) & \boldsymbol{k} \end{bmatrix} \tag{13.30}$$

$$Q_1 Q_2 Q_3 Q_4 Q_5 Q_6 L_2 \widetilde{Q}_6 \widetilde{Q}_5 \widetilde{Q}_4 \widetilde{Q}_3 \widetilde{Q}_2 \widetilde{Q}_1 =$$

$$\begin{bmatrix} L_3(s_2 c_3 c_5 + c_2 s_4 s_5 - s_2 s_3 c_4 s_5) & \boldsymbol{i} \\ L_3(s_1 s_3 c_5 - c_1 c_2 c_3 c_5 + c_1 c_2 s_3 c_4 s_5 & \\ \quad + c_1 s_2 s_4 s_5 + s_1 c_3 c_4 s_5) & \boldsymbol{j} \\ L_3(s_1 s_2 s_4 s_5 - c_1 s_3 c_5 - s_1 c_2 c_3 c_5 & \\ \quad - c_1 c_3 c_4 s_5 + s_1 c_2 s_3 c_4 s_5) & \boldsymbol{k} \end{bmatrix}. \tag{13.31}$$

13.5.2 The Jacobian

The forward kinematics can be written as an exponential mapping [15]. Thus, the forward kinematics of an inverted pendulum can be defined as $p' = e^{\theta s} p$, where s represents the screw or Lie algebra in the joint and $p \in \mathbb{PR}^3$. Then, the forward kinematics for a robot with any number of DOF can be illustrated as (13.2).

$$p' = K(\theta)p \quad \text{where} \quad K(\theta)\prod_{j=1}^{n} e^{\theta_j s_j}, \tag{13.32}$$

where s_j represents the Lie algebra for each DOF. Hence, using (13.32), the velocity of the end-effector can be determined by (13.33)

$$\dot{p'} = \frac{d}{dt}K(\theta)p, \tag{13.33}$$

therefore

$$\dot{p'} = \left(\sum_{j=1}^{n} \dot{\theta}_j S_j\right) p \quad \text{where} \quad S_j = \frac{\partial K}{\partial \theta_j}. \tag{13.34}$$

Commonly, in robotics, the velocity of the end-effector is written as

$$\begin{pmatrix} w \\ v \end{pmatrix} = \begin{pmatrix} J_w \\ J_v \end{pmatrix}\dot{\theta} = J\dot{\theta} \quad \text{where} \quad \dot{\theta}^T = [\dot{\theta}_1, \dot{\theta}_2, \ldots, \dot{\theta}_n]^T, \tag{13.35}$$

where w is the angular velocity, v is the linear velocity, J is the robot Jacobian, J_w is the Jacobian of angular velocities, J_v is the Jacobian of linear velocities, and $\dot{\theta}$ is the vector of joint velocities. Therefore, the robot Jacobian can be represented as

$$J = (S_1, S_2, \ldots, S_n). \tag{13.36}$$

According to the foregoing, using quaternions, the screw can be defined as

$$S_j = \begin{bmatrix} v_j \\ \frac{1}{2}\sum_{l=1}^{L}(R(\boldsymbol{v_n}R(L_l)|_n^{j_l})|_1^{n-1} - R(R(L_l)|_n^{j_l}\boldsymbol{v_n})|_1^{n-1}) \end{bmatrix}, \tag{13.37}$$

where L is the number of joints or points to rotate, L_l refers to the l-th articulation, $R(\cdot)|_n^{j_l}$ refers to the kinematic chain from n to a_l, i.e., $Q|_a^b = Q_a Q_{a+1} \ldots Q_b$ and $\widetilde{Q}|_b^a = Q_b Q_{b-1} \ldots Q_a$ (obviously the rotation $R(\cdot)|_n^{a_l}$ only exists if $n \leq a_l$), and $\boldsymbol{v_1} = i$, $\boldsymbol{v_2} = k$, $\boldsymbol{v_3} = i$, $\boldsymbol{v_4} = -j$, $\boldsymbol{v_5} = i$, $\boldsymbol{v_6} = -j$. In the same way, $R(\cdot)|_1^{n-1}$ is defined.

In base with Eq. (13.37), the Jacobian for the MexOne arm can be constructed by

$$S_1 = \begin{bmatrix} v_1 \\ \frac{1}{2}(\boldsymbol{v_1} Q|_1^2 L_1 \widetilde{Q|_2^1} + \boldsymbol{v_1} Q|_1^4 L_2 \widetilde{Q|_4^1} \cdots \\ +\boldsymbol{v_1} Q|_1^6 L_3 \widetilde{Q|_6^1} - (Q|_1^2 L_1 \widetilde{Q|_2^1} \boldsymbol{v_1} \cdots \\ +Q|_1^4 L_2 \widetilde{Q|_4^1} \boldsymbol{v_1} + Q|_1^6 L_3 \widetilde{Q|_6^1} \boldsymbol{v_1})) \end{bmatrix},$$

$$S_2 = \begin{bmatrix} v_2 \\ \frac{1}{2}(Q_1 \boldsymbol{v_2} Q_2 L_1 \widetilde{Q_2^1} + Q_1 \boldsymbol{v_2} Q|_2^4 L_2 \widetilde{Q|_4^1} \\ +Q_1 \boldsymbol{v_2} \widetilde{Q|_2^6} L_3 \widetilde{Q|_6^1} - (Q|_1^2 L_1 \widetilde{Q_2} \boldsymbol{v_2} Q_1 \\ +Q|_1^4 L_2 \widetilde{Q|_4^2} \boldsymbol{v_2} Q_1 + Q|_1^6 L_3 \widetilde{Q|_6^2} \boldsymbol{v_2} Q_1)) \end{bmatrix},$$

$$S_3 = \begin{bmatrix} v_3 \\ \frac{1}{2}(Q_1^2 \boldsymbol{v_3} Q|_3^4 L_2 \widetilde{Q|_4^1} + Q_1^2 \boldsymbol{v_3} Q|_3^6 L_3 \widetilde{Q|_6^1} \\ -(Q|_1^4 L_2 \widetilde{Q|_4^3} \boldsymbol{v_3} Q_2^1 + Q|_1^6 L_3 \widetilde{Q|_6^3} \boldsymbol{v_3} Q_2^1)) \end{bmatrix},$$

$$S_4 = \begin{bmatrix} v_4 \\ \frac{1}{2}(Q_1^3 \boldsymbol{v_4} Q_4 L_2 \widetilde{Q|_4^1} + Q_1^3 \boldsymbol{v_4} Q|_4^6 L_3 \widetilde{Q|_6^1} \\ -(Q|_1^4 L_2 \widetilde{Q_4} \boldsymbol{v_4} Q_3^1 + Q|_1^6 L_3 \widetilde{Q|_6^4} \boldsymbol{v_4} Q_3^1)) \end{bmatrix},$$

$$S_5 = \begin{bmatrix} v_5 \\ \frac{1}{2}(Q_1^4 \boldsymbol{v_5} Q_5^6 L_3 \widetilde{Q|_6^1} \\ -(Q|_1^6 L_3 \widetilde{Q|_6^5} \boldsymbol{v_5} Q_4^1)) \end{bmatrix},$$

$$S_6 = \begin{bmatrix} v_6 \\ \frac{1}{2}(Q_1^5 \boldsymbol{v_6} Q_6 L_3 \widetilde{Q|_6^1} \\ -(Q|_1^6 L_3 \widetilde{Q|_6} \boldsymbol{v_6} Q_5^1)) \end{bmatrix}. \tag{13.38}$$

Therefore, developing the previous equations, the screws can be written as

$$S_1 = \begin{bmatrix} 1 & \boldsymbol{i} \\ 0 & \boldsymbol{j} \\ 0 & \boldsymbol{k} \\ 0 & \boldsymbol{i} \\ L3(c_1 s_3 c_5 - s_1 c_2 c_3 c_5 - s_1 c_2 s_3 c_4 s_5 - s_5 s_4 s_2 s_1 & \\ +s_5 c_4 c_3 c_1) + L_2(c_1 s_3 + s_1 c_2 c_3) + L1 s_1 c_2 & \boldsymbol{j} \\ L3(s_5 s_4 s_2 c_1 + c_5 s_3 s_1 - c_5 c_3 c_2 c_1 + s_5 c_4 c_3 s_1 & \\ +s_5 c_4 s_3 c_2 c_1) + L2(s_1 s_3 - c_1 c_2 c_3)) - L_1 c_1 c_2 & \boldsymbol{k} \end{bmatrix},$$

$$S_2 = \begin{bmatrix} 0 & \boldsymbol{i} \\ 0 & \boldsymbol{j} \\ 1 & \boldsymbol{k} \\ L_3(c_2c_3c_5 - s_2s_4s_5 - c_2s_3c_4s_5) + & \\ L_2c_2c_3 + L_1c_2 & \boldsymbol{i} \\ L_3(c_1s_2c_3c_5 - c_1s_2s_3c_4s_5 + s_5s_4c_2c_1) & \\ +L_2c_1s_2c_3 + L_1c_1s_2 & \boldsymbol{j} \\ L_3(s_5s_4c_2s_1 + c_5c_3s_2s_1 - s_5c_4s_3s_2s_1) + & \\ L_2s_1s_2c_3 + L_1s_1s_2 & \boldsymbol{k} \end{bmatrix},$$

$$S_3 = \begin{bmatrix} 1 & \boldsymbol{i} \\ 0 & \boldsymbol{j} \\ 0 & \boldsymbol{k} \\ -L_3(s_2s_3c_5 + s_2c_3c_4s_5) - L_2s_2s_3 & \boldsymbol{i} \\ L_3(s_1c_3c_5 + c_1c_2s_3c_5 + c_1c_2c_3c_4s_5 - & \\ s_5c_4s_3s_1) + L_2c_1c_2s_3 & \boldsymbol{j} \\ L_3(c_5s_3c_2s_1 - c_5c_3c_1 + s_5c_4s_3c_1 + & \\ s_5c_4c_3c_2s_1) - L_2(c_1c_3 - s_1c_2s_3) & \boldsymbol{k} \end{bmatrix},$$

$$S_4 = \begin{bmatrix} 0 & \boldsymbol{i} \\ -1 & \boldsymbol{j} \\ 0 & \boldsymbol{k} \\ L_3(c_2c_4s_5 + s_2s_3s_4s_5) & \boldsymbol{i} \\ L_3(s_5c_4s_2c_1 - c_1c_2s_3s_4s_5 - s_5s_4c_3s_1) & \boldsymbol{j} \\ L_3(s_5c_4s_2s_1 + s_5s_4c_3c_1 - s_5s_4s_3c_2s_1) & \boldsymbol{k} \end{bmatrix},$$

$$S_5 = \begin{bmatrix} 1 & \boldsymbol{i} \\ 0 & \boldsymbol{j} \\ 0 & \boldsymbol{k} \\ L_3(c_2s_4c_5 - s_2c_3s_5 - s_2s_3c_4c_5) & \boldsymbol{i} \\ L_3(c_5c_4c_3s_1 + c_5s_4s_2c_1 + c_1c_2s_3c_4c_5 + & \\ c_1c_2c_3s_5 - s_1s_3s_5) & \boldsymbol{j} \\ L_3(s_5s_4s_2s_1 + s_5s_3c_1 + s_5c_3c_2s_1 - & \\ c_5c_4c_3c_1 + c_5c_4s_3c_2s_1) & \boldsymbol{k} \end{bmatrix},$$

$$S_6 = \begin{bmatrix} 0 & \boldsymbol{i} \\ -1 & \boldsymbol{j} \\ 0 & \boldsymbol{k} \\ 0 & \boldsymbol{i} \\ 0 & \boldsymbol{j} \\ 0 & \boldsymbol{k} \end{bmatrix}.$$

Thus, the Jacobian is represented as

$$J = (S_1 \quad S_2 \quad S_3 \quad S_4 \quad S_5 \quad S_6). \tag{13.39}$$

13.5.3 Inverse Kinematics

The definition of the inverse kinematics can be represented as

$$\theta = f(E), \tag{13.40}$$

where θ is the joint space and E is the Euclidean space. In other words, the inverse kinematics is the technique that obtains the necessary angles to reach a random position. The inverse kinematics is not a trivial technique, and sometimes it is impossible to find a solution; additionally, the inverse kinematics depends on the DOF of the robot. So, to avoid specific solutions, the following theorem is proposed.

Theorem 1 *The inverse kinematics of any robot can be obtained by (13.41), where an extra necessary condition is that the desired trajectory is smooth*

$$\theta = \varepsilon \int_{t_0}^{t} {J_v}^T e \quad dt, \tag{13.41}$$

where $\theta \in \mathbb{R}^n$, $\varepsilon \in \mathbb{R} > 0$, and e is the error between the desired quaternion and the measured quaternion. To prove the effectiveness of the theorem, the stability of the error is necessary. So, the following Lyapunov function is suggested:

$$V(e) = \frac{\varepsilon}{2} e^T e. \tag{13.42}$$

Applying the derivative with respect to time of the Lyapunov function:

$$\dot{V(e)} = \frac{\varepsilon}{2} (\dot{e^T} e + e^T \dot{e}). \tag{13.43}$$

The error is defined as the projection between the desired quaternion and the measured; therefore, its derivative with respect to time can be represented as $\dot{e} = -J_v \dot{\theta}$. So, (13.43) changes to (13.44).

$$\dot{V(e)} = -\frac{\varepsilon}{2} ((J_v \dot{\theta})^T e + e^T J_v \dot{\theta}). \tag{13.44}$$

To satisfy the stability, $\dot{\theta}$ is taken as a control signal. Therefore, if $\dot{\theta} = {J_v}^T e$, (13.44) can be written as

$$\dot{V(e)} = -\frac{\varepsilon}{2} ((J_v {J_v}^T e)^T e + e^T J_v {J_v}^T e), \tag{13.45}$$

$$\dot{V(e)} = -\varepsilon e^T J_v {J_v}^T e. \tag{13.46}$$

So, with $\dot{\theta} = {J_v}^T e$ and $\varepsilon \in \mathbb{R} > 0$, the derivative of the Lyapunov function is negative definite, and the system is globally asymptotically stable [16]. Therefore, the solution of the inverse kinematics with Theorem 1 is valid.

13.5.4 Control

In robotics, the control is an important topic to cover the requirements of robustness of robots. Traditionally, techniques as PD, PID, PD+G, and CTC are the most trendy controls used in robotics, but the mean problem with these controllers is that the dynamic equations of the robot should be as accurate as possible. Another problem is the possibility that an external load is presented in the robot, and this load changes the dynamic of the robot and the controllers need to be reconfigured [17]. The solution of these problems is to use an adaptive controller, where the neural networks can be used as adaptive controllers and offer good results in robotics [4].

The design of the controller in this work can be seen in Fig. 13.6, and this controller is a model of spiking neural networks with a quaternionic expansion. The model has three neurons in its first layer and these neurons receive the robot position, the robot velocity, and the desired position to reach. On the other hand, the second layer and third layer have one neuron, respectively, and these neurons receive three signals of the first layer plus the output of the previous neuron. So, the control signal will be the output of the neural network, where this signal will be the torque that the robot needs to reach the desired position.

Control of a robot arm. *Point-to-point control*

The point-to-point control was performed with a desired arbitrary constant reference, located at the coordinates $[10, -10, 20]_{cm}$. Figure 13.12a shows the trajectory from the home position (blue lines) to the final position (red lines). Figure 13.12b describes the evolution of the angles and it can be seen that after two seconds all the angles converge, and this means that the final effector reached the desired reference. Finally, in Fig. 13.12c, the mean square error is described, which converges to zero at the same time that all the angles become constant.

Control for function tracking

To prove the effectiveness of the algorithm, two examples are illustrated. In both examples, the MexOne arm is the robot considered to reach the desired positions. The first example shows the MexOne arm tracking the Lemniscate function, and this function can be represented in Eq. (13.47).

$$\dot{V(e)} = -\varepsilon e^T J_v {J_v}^T e, \tag{13.47}$$

with e^T equal to

$$\begin{bmatrix} (\cos(t)' * 17 + 13) - X_f(t) \\ (\sin d(t * 2)' * 17 + 13) - Y_f(t) \\ 19 - Z_f(t) \end{bmatrix}, \tag{13.48}$$

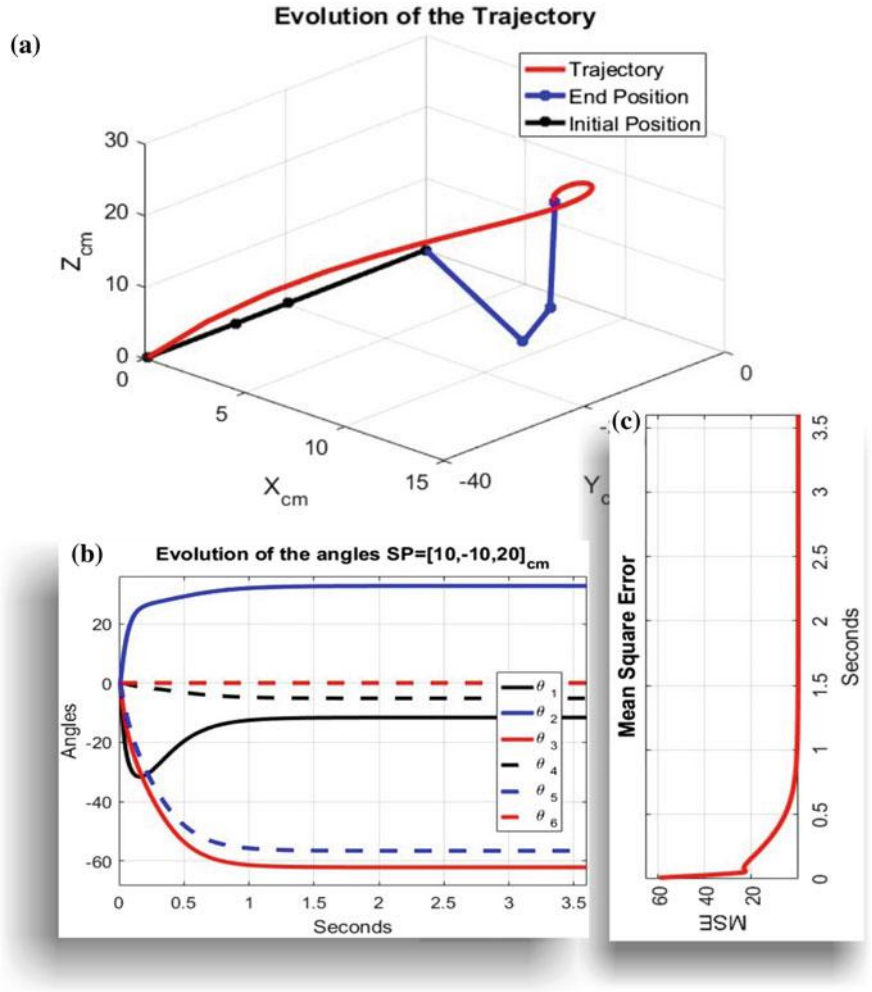


Fig. 13.12 **a** The behavior of the manipulator of six degrees of freedom, from its initial position to the final position. **b** The evolution of the angles through time. **c** The error dynamics

where X_f, Y_f, and Z_f are the coordinates of the end-effector. The results of the algorithm can be seen in Figs. 13.13, 13.14, and 13.15, respectively.

On the other hand, the second example illustrates the MexOne arm tracking the Rhodonea function, where this function can be seen in Eq. (13.49)

$$\dot{V(e)} = -\varepsilon e^T J_v {J_v}^T e, \tag{13.49}$$

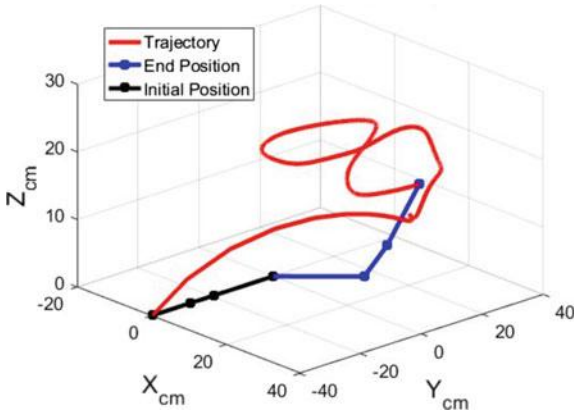


Fig. 13.13 Evolution of the robot position using the Lemniscate function

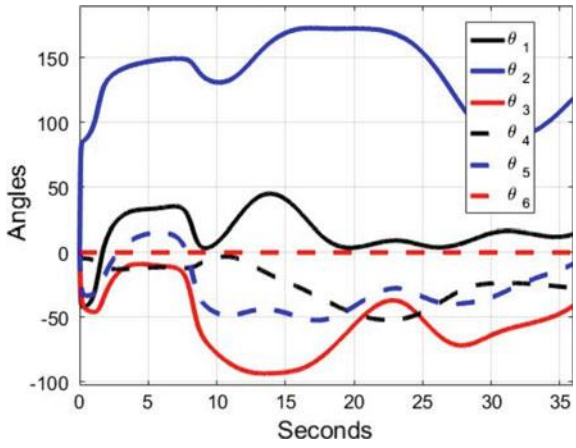


Fig. 13.14 Behavior of the joint angles using the Lemniscate function

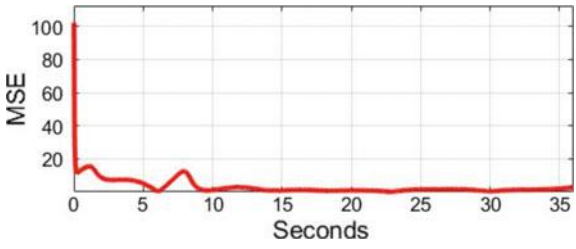


Fig. 13.15 Convergence error using the Lemniscate function

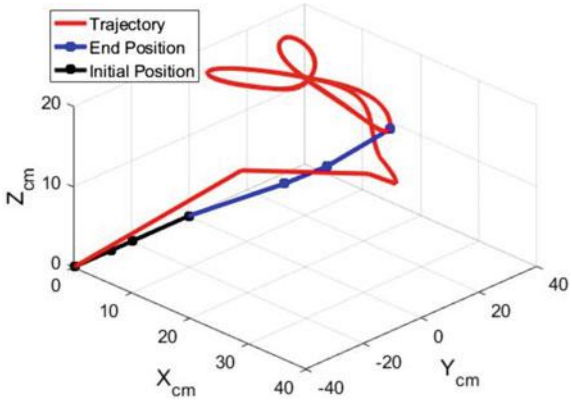


Fig. 13.16 Evolution of the robot position using the Rhodonea function

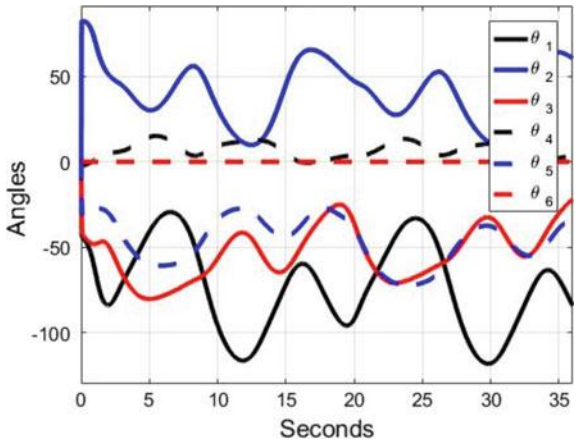


Fig. 13.17 Behavior of the joint angles using the Rhodonea function

with e^T equal to

$$\begin{bmatrix} (\cos d(t*3)*\cos(t)*17+13)-X_f(t) \\ (\cos(t*3)*\sin(t)'*17+13)-Y_f(t) \\ 19-Z_f(t) \end{bmatrix}, \tag{13.50}$$

where the results of the algorithm can be seen in Figs. 13.16, 13.17, and 13.18, respectively.

13.6 Neural Network Signal Processing

Neural networks are utilized in different areas of engineering, due to its great ability to learn time-variant nonlinear mappings according to system properties. The QSNN is a powerful neural network which can both identify a plant and simultaneously control it.

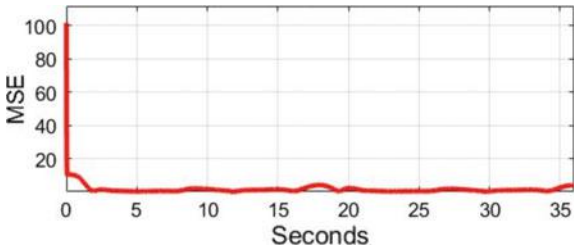


Fig. 13.18 Convergence error using the Rhodonea function

Fig. 13.19 The robotic hand prosthesis

We believe that QQNN acts as direct controller like an adaptive PID controller. Moreover, we suspect that in biological creatures the neural networks mimic adaptive PID controllers, because this kind of controller is simple and very efficient and the nature should have developed such control models through evolution.

The purpose of using spike-type quaternion neural networks to process information is to be able to complement the project of a robotic prosthesis in order to manipulate the information coming from the forearm and with this information we can be able to manipulate and control the movements of the handheld robotic prosthesis, Fig. 13.19.

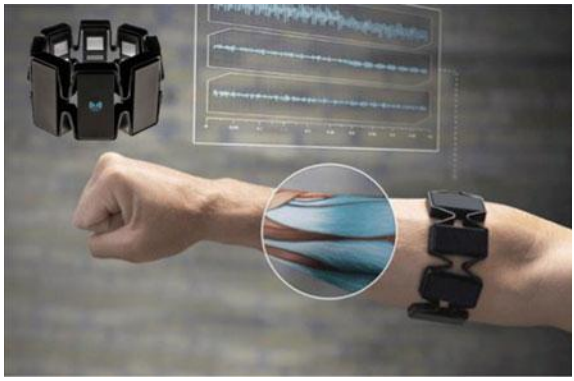


Fig. 13.20 Myo bracelet type: Myo Armband—Black

13.6.1 Myo Bracelet

The device is called Myo and is composed of a set of links that integrate high-sensitivity medical-grade stainless steel sensors. This device is able to recognize and interpret the electrical activity of the muscles, generating eight signals that come from the perimeter muscles located in the forearm; in the same way, three gyroscopic signals are generated (pitch, roll, and yaw) to be able to calculate if necessary the orientation of the bracelet; Fig. 13.20 shows the Myo device implemented to obtain a representation of the perimeter myoelectric signals of the muscles of the forearm, and among them are external radial muscle of the forearm, long palm muscle, muscle superficial flexor, and brachial radial muscle.

13.6.2 Physiology of Robotic Prosthesis

Figure 13.21 shows the internal structure of the limb of an upper limb, from the metacarpals to the distal phalanges. By making a simplified representation of the kinematic chain of the structure of a finger, considering each bone joint as a joint of a single degree of freedom, the equivalent diagram of Fig. 13.22 is obtained.

This diagram is constituted by four degrees of freedom (θ_1, θ_2, θ_3, θ_4); in the case of the thumb three, this configuration does not correspond completely to the complex locomotion of the finger, but nevertheless it has been decided to simplify it even more, since that an open mechanism with more than one link will always have more than one degree of freedom, so it requires as many actuators (motors) as there are degrees of freedom, Franz Reuleaux (1829–1905) [19].

The latter compromises the ergonomy of the design of the prosthesis, since in addition to the four actuators that would be included, its respective elements of measurement and operation must also be implemented.

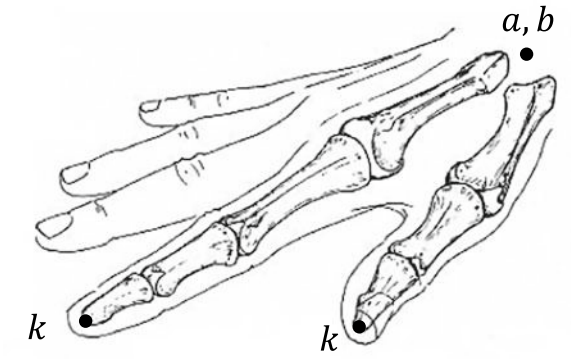


Fig. 13.21 Bone system of finger and thumb of the human hand [18]

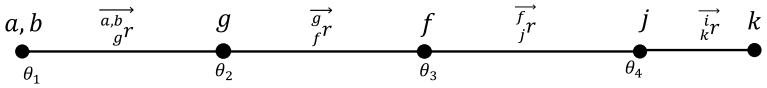


Fig. 13.22 Open equivalent kinematic chain of a finger

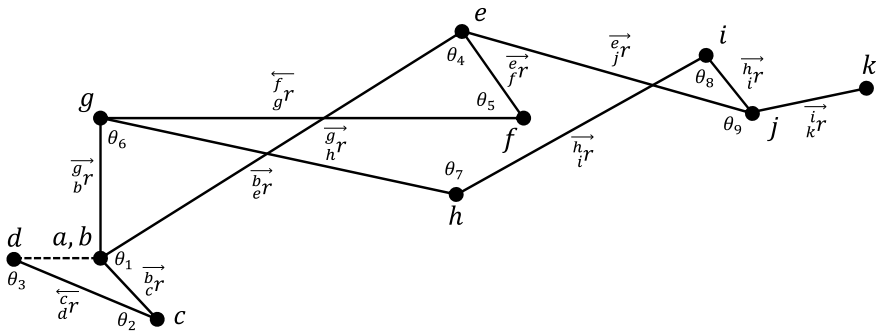


Fig. 13.23 Closed equivalent kinematic chain of a finger

In view of this, the Grbler–Kutzbach criterion [20] is applied, which determines the degree of freedom (mobility) of a kinematic chain, that is, a coupling of rigid bodies by means of mechanical constraints

$$M = 3(L - 1) - 2J_1 - J_2, \tag{13.51}$$

where M are the degrees of freedom, L number of links, J_1 number of joints of complete, and J_2 number of semi-joints.

The resulting chain is a single degree of freedom shown in Fig. 13.23, and this chain is formed by seven links, eight together and a semi-joint, which means the implementation of a single actuator, and before this configuration the chain becomes a restricted system Franz Reuleaux (1829–1905) [20].

Table 13.1 Possible cases in mechanical closing systems

Case 1	$C, \widehat{C}$
Case 2a	A, B
Case 2b	$A, \widehat{B}$
Case 2c	$\widehat{A}, \widehat{B}$

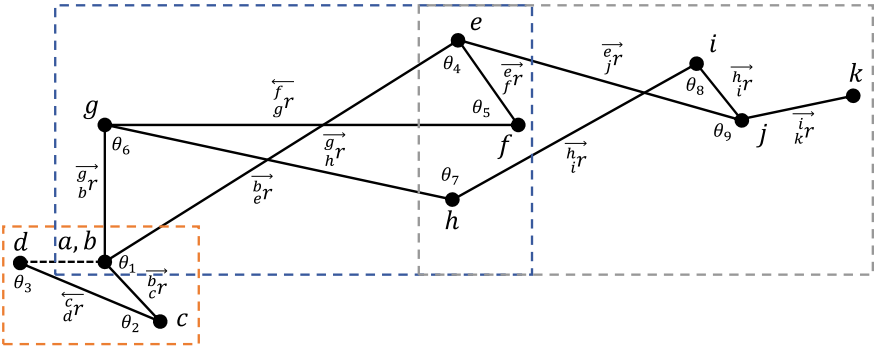


Fig. 13.24 Internal closed kinematic chain of the restricted system

The above criteria only works for the calculation of mobility of the system, and the lengths of each of the seven links that make up the chain must still be determined, and for this the method of Milton A. Chace (1967–1985) [20] is applied, namely, the vector equation of closure in the plane takes advantage of the brevity of vector notation in obtaining explicit solutions in closed form.

A two-dimensional vector equation can be solved for two unknowns: two magnitudes, two directions or a magnitude and an address, starting from the typical vector equation in the following plane equation:

$$C\widehat{C} = A\widehat{A} + B\widehat{B}. \tag{13.52}$$

To solve the above equation, four possible cases are shown in Table 13.1.

The restricted system in question consists of three restricted closed chains, Fig. 13.24.

All the internal chains correspond to case 2c of the Chace method, Table 13.1, regardless of the assembly configuration the three internal chains are equivalent to the model of a four-bar system, Fig. 13.25, where Eqs. (13.53) and (13.54) calculate two possible symmetric results for the model.

$$r_3 = \pm\sqrt{r_3^2 - \left(\frac{r_3^2 - r_4^2 + s^2}{2s}\right)^2}\,(\widehat{s} \times \widehat{k}) + \frac{r_3^2 - r_4^2 + s^2}{2s}\widehat{s} \tag{13.53}$$

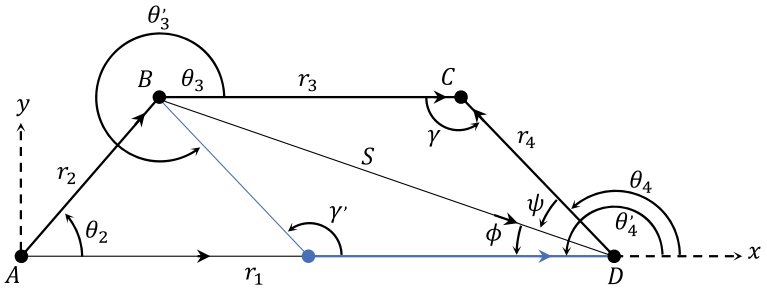


Fig. 13.25 Restricted kinematic chain of a four-bar mechanism

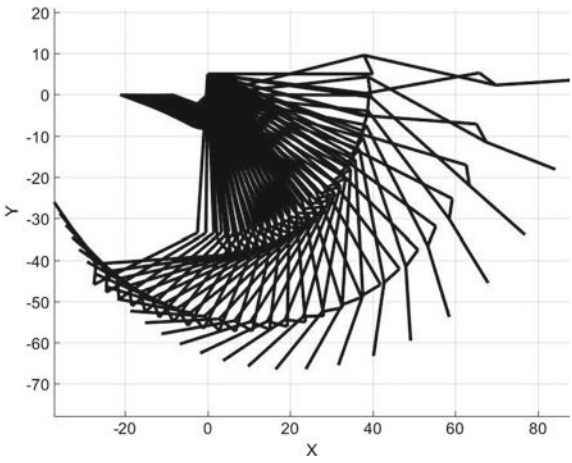


Fig. 13.26 Complete kinematics from extension to flexion, metric system (mm)

$$r_4 = \pm\sqrt{r_3^2 - \left(\frac{r_3^2 - r_4^2 + s^2}{2s}\right)^2}(\widehat{s} \times \widehat{k}) + \frac{r_4^2 - r_3^2 + s^2}{2s}\widehat{s}. \tag{13.54}$$

Once the closing equations of the whole restricted system have been obtained, they are established as an objective function for the implementation of the optimization algorithm by **PSO** (optimization by swarm of particles) [21], because there are too many variables that make up the system, and this algorithm optimization allows the heuristic calculation of a swarm of particles with 24 dimensions each, dimensions that correspond to the planar units of each link. In addition, given the biological morphology of the human finger, the objective function is bounded under the Fibonacci ratio for the phalanges [22], and Fig. 13.26 shows the restricted complete kinematics.

13.6.3 Preprocessing and Training

In order to implement the pulse-type quaternion artificial neural networks in the processing of myoelectric signals, it is necessary to carry out several stages.

Figure 13.27 shows the first stage (the training), where before starting with the neuronal training, the eight signals (emitted by the Myo bracelet) are divided into two groups of four, this in order to arm a pair of quaternions, one for the "top" and the other quaternion would process the signals from the bottom of the bracelet. Each of these signals must be first preprocessed to perform this operation and they are introduced to an array of quaternionic neural networks, but with the characteristic that said signals are not affected by neuronal weights, which means that they are only used to the QSN as a kind of low-pass filters, and this is done in order to eliminate both noise and frequencies in which the spike neural networks are not able to work.

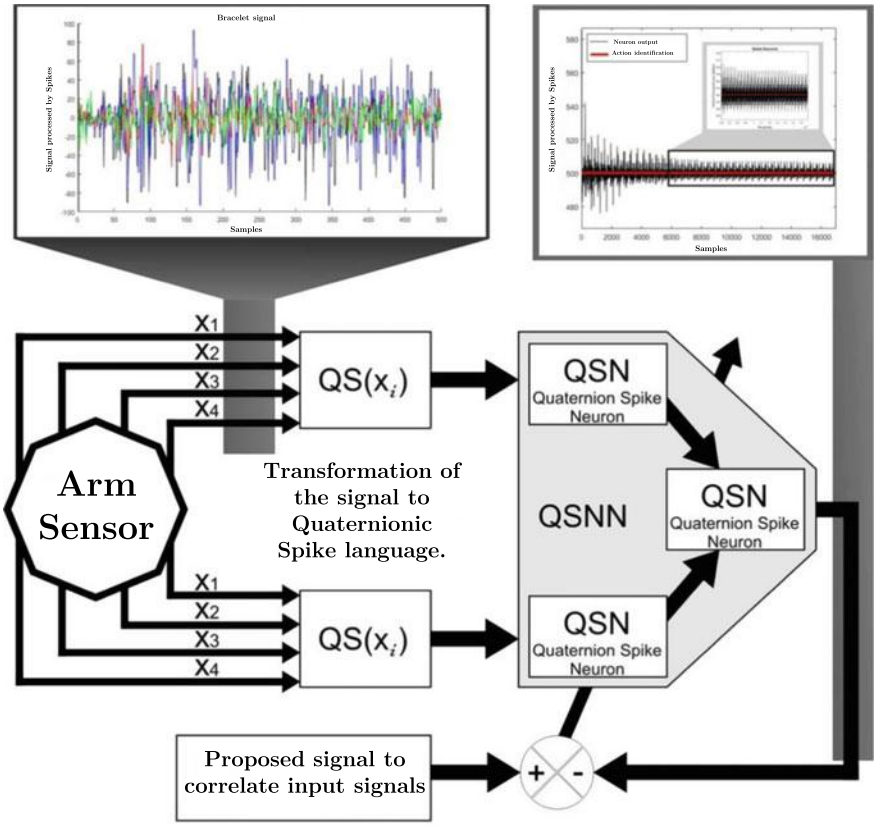


Fig. 13.27 Training scheme for myoelectronic signals of Myo bracelet

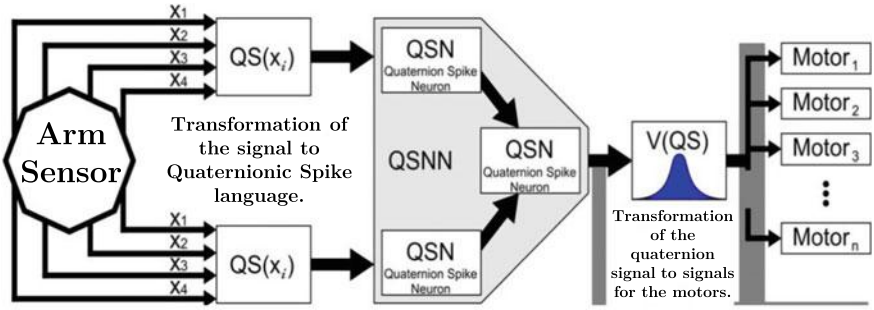


Fig. 13.28 Diagram of identification and control for myoelectronic signals of wristband Myo

The two quaternions that are obtained from the preprocessing are introduced to the neural network to be able to be trained with this information. The architecture used is a feedforward type with two layers with two neurons in the input layer and one in the output layer, and the spike-type quaternionic neural networks are systems that respond to supervised training, so there must be an error for that the training is done (adjustment of weights), and in order to generate an error it is necessary to have (or propose) a desired output which will be compared with the output generated by the network. Because the purpose of the artificial networks for this work is to identify different sets of signals to be later associated with movements in the robotic prosthesis, it is proposed that the output for each training be just half in the neural time window.

13.6.4 Evaluation and Control

Once the neural network is trained, i.e., the weights converge to a region through the training algorithm, it is able to identify each of the positions trained. So you should move on to the next stage, Fig. 13.28.

The purpose of training on any neural network is to create a function to transform the inputs into outputs that follow the restrictions generated by the function that depends on the inputs themselves. Therefore, in this second stage, what the neural network receives as trained has to go through the same transformations applied in the training stage, and this means that, in the same way, in the first stage the eight entries are divided into two groups and are transformed to eliminate the frequencies that the QSNN cannot process. Once generated, two quaternions are introduced as input to the neural network that will adjust according to the weights obtained in the training. The output is evaluated in

$$Out(t_{QN}) = e^{-\frac{\left(t_{QN}-\frac{tN}{2}\right)^2}{\frac{tN^2}{25}}}, \tag{13.55}$$

where t_{QN} is the neural output and tN is the final value of the neural time window. Since the Gaussian function is not scaled, the response sought is in the range [1] and each trained position is evaluated in the Gaussian equation (13.55).

13.7 Control of a Hand Prosthesis

The training was carried out for the identification of five states (relax, close, open, point, and thumb of approval) of which seven thousand samples of each one were obtained in order to train the neural network.

Complete description of the identification of signals. To give a complete explanation, the entire recognition algorithm for the first 130 samples is analyzed.

The eight input signals in Fig. 13.29 are divided into two quaternions assigning each signal as if it were a quaternion, once having a pair of quaternions are introduced to a spike-type neuron without weight association or law of learning, to be able to filter the input signal using the same neurons, Fig. 13.30.

Each of the signals in Fig. 13.30 is associated (the neural network is trained) with the three signals generated by the gyroscope, Fig. 13.31, in this way, the neural network creates a three-dimensional function that depends on the pitch, yaw, and roll variables. The implemented neural network uses spike quaternionic neurons with a LIF-type neuronal model which has a resistance of 330 Ω, a capacitance of 20 F, with a number of delays equal to 100 and each delay associated with a time of 3 ms. The training implemented is the Widrow–Hoff rule extended to quaternions.

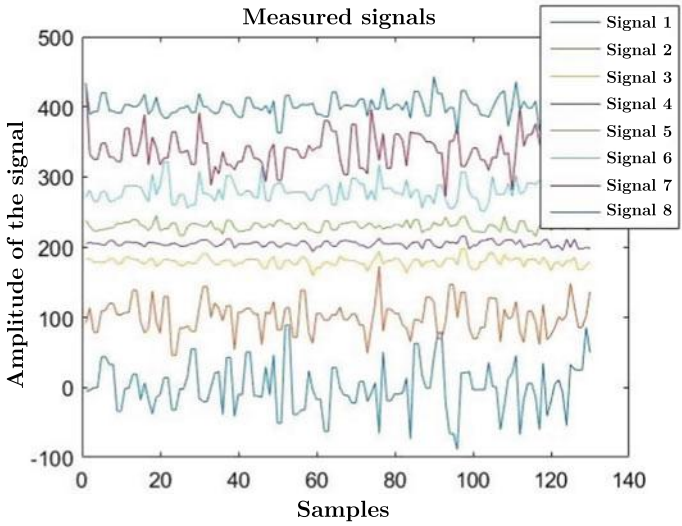


Fig. 13.29 Eight sensed signals from the Myo bracelet

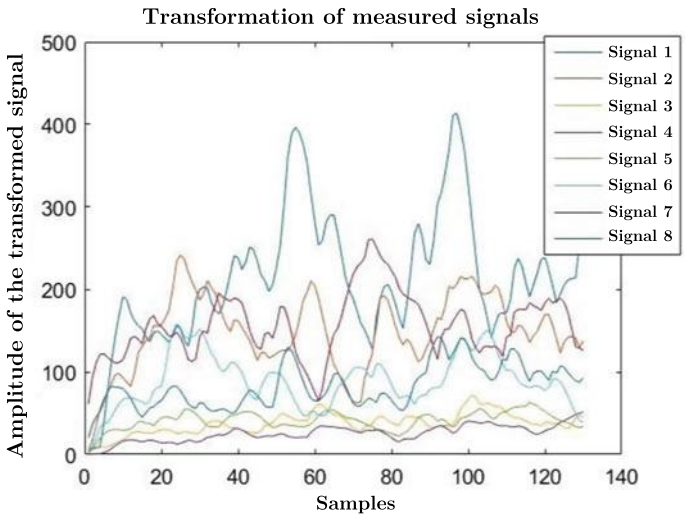


Fig. 13.30 Result of signal filtering with spike neural networks

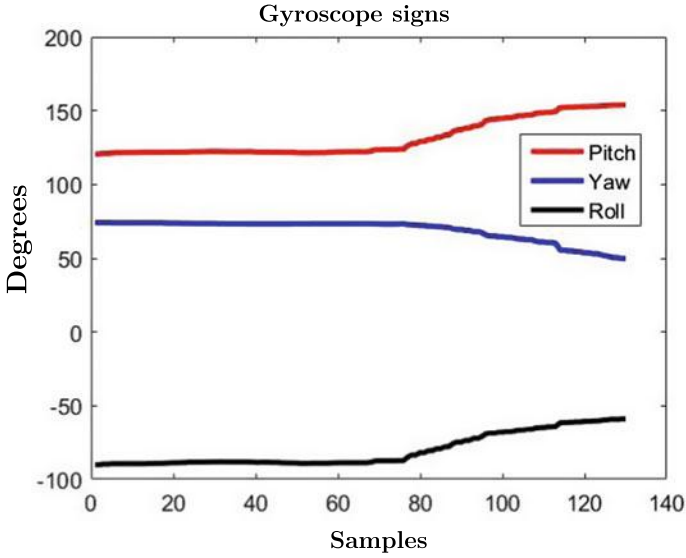


Fig. 13.31 Three sensed gyro signals from the Myo bracelet

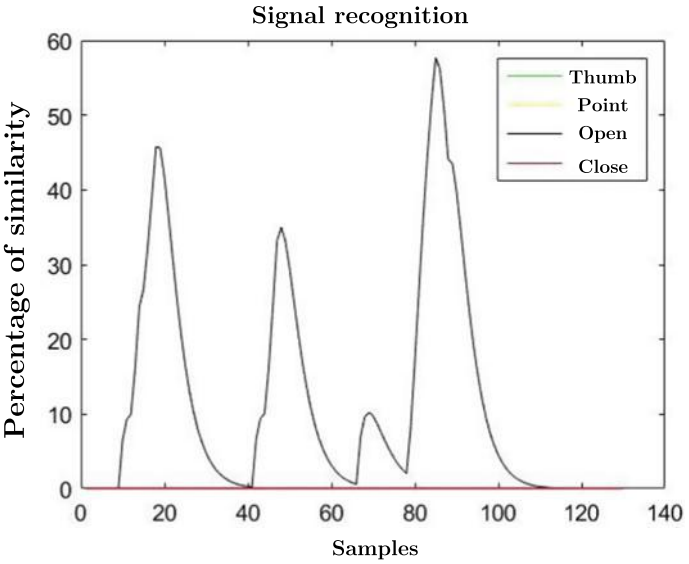


Fig. 13.32 Recognition of the sensed signals of the Myo bracelet

Once the quaternionic neural network is trained, the architecture is able to recognize which signal it is. This task can be performed because the three signals of the gyroscope enter the trained quaternionic neural network and the output is compared with the reading of the quaternions generating a percentage of similarity. Figure 13.32 shows the result of the whole identification process within the first 130 samples of a new signal sampled (open hand). In this image, it is seen that the algorithm defines the signal sensed by the Myo (with a probability of up to 58%) as the action of opening.

Once having the recognition of how the robotic prosthesis should act, the signal is sent to activate one of the cases to move the actuators and reach the predetermined position.

Final Check. Once the training process was completed, it was proposed to use a confusion matrix to evaluate the implementation of the quaternion spike neural networks. Figure 13.33 shows how the identification of each state is tested and compared against all the other, i.e., each column corresponds to a state and in each row the evaluation of each state. In general, it can be seen that there is not a single misinterpreted moment in the identification and there are only events in the main diagonal of the confusion matrix.

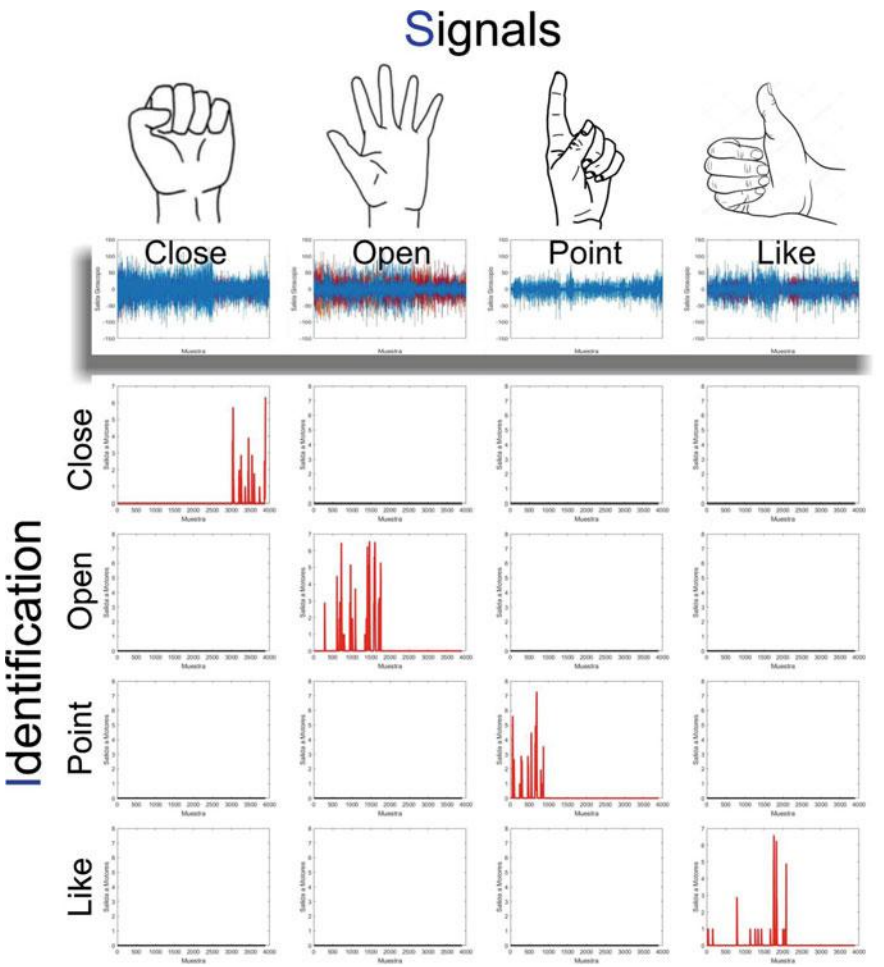


Fig. 13.33 The confusion matrix to the identification of each state

13.8 Conclusions

Quaternion spiking neural networks are able to control any robot, where the examples confirm that these artificial neurons have the capacity to adapt the robot to reach the desired position. In the first application, the diverse desired trajectories are shown, where it is possible to see the effectiveness of the technique. But, these examples do not have perturbations; therefore, the future work is to prove the robustness of the algorithm with or without external perturbations.

References

1. Izhikevich, E. M. (2003). Simple model of spiking neurons. *IEEE Transactions on Neural Networks*, *14*(6), 1569–1572.
2. Maass, W. (1997). Networks of spiking neurons: The third generation of neural network models. *IEEE Transactions on Neural Networks*, *10*(9), 1659–1671.
3. Abiyev, R. H., Kaynak, O., & Oniz, Y. (2012). Spiking neural networks for identification and control of dynamic plants. In *International Conference on Advanced Intelligent Mechatronics*.
4. Lechuga-Gutiérrez, L. R., & Bayro-Corrochano, E. (2016). Quaternion spike neural networks. In *International Symposium on Neural Networks (ISNN)* (Vol. 9719, pp. 640–646).
5. Hodgkin, A. L., & Huxley, A. F. (1952). A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of Physiology*, *9*(4), 500–544.
6. Thorpe, S., Delorme, A., & Van Rullen, R. (2001). Spike-based strategies for rapid processing. *Neural Networks, 14*(4), 715–725 (Elsevier).
7. Van Rullen, R., Delorme, A., Gautrais, J., & Thorpe, S. (1999). SpikeNET: A simulator for modeling large networks of integrate and fire neurons. *Neurocomputing*, *26–27*(8), 989–996.
8. Gerstner, W., & Kistler, W. M. (2002). *Spiking neuron models, single neurons, population, plasticity*. Cambridge: MIT Press.
9. Izhikevich, E. M. (2004). Which model to use for cortical spiking neurons? *IEEE Transactions on Neural Networks*, 1063–1070.
10. NatschlNager, T., & Ruf, B. (1998). Spatial and temporal pattern analysis via spiking neurons. *Network: Computation in Neural Systems, 9*(3), 319–332.
11. Bohte, S., Kok, J., & Han, L. (2001). Error-back propagation in temporally encoded networks of spiking neurons. *Network: Computation in Neural Systems, 1*(48), 17–37.
12. Haykin, S. (1999). *Neural networks: A comprehensive foundation*. New York: Prentice-Hall.
13. Mohemmed, A., Schliebs, S., Matsuda, S., & Kasabov, N. (2011). Method for training a spiking neuron to associate input-output spike trains. In *International Conference on Engineering Applications of Neural Networks IFIP International Conference on Artificial Intelligence Applications and Innovations EANN 2011*. AIAI 2011: Engineering Applications of Neural Networks (Vol. 363, pp. 219–228).
14. Spong, M. W., Hutchinson, S., & Vidyasagar, M. (2006). *Robot modeling and control*. New York: Wiley.
15. Selig, J. (1992). *Introductory robotics*. Hertfordshire, UK: Prentice Hall International.
16. Khalil, H. K. (2002). *Nonlinear systems*. East Lansing: Pearson.
17. Shao, Z., & Zhang, X. (2010). Intelligent control of flexible-joint manipulator based on singular perturbation. In *IEEE International Conference on Automation and Logistics (ICAL)*.
18. Pró, E. A. (2012). Anatomía clínica pró. Editorial Panamericana.
19. Norton, R. L. (2009). *Machine design*. McGraw-Hill.
20. Shigley, J. E., Uicker, J. J., & de Contin, H. C. (1996). *Theory of mechanisms and machines*. McGraw-Hill.
21. Cuevas Jiménez, E. V., & Osuna Enciso, J. V. (2016). Optimización algoritmos programados con matlab. Alfaomega
22. Kapandji, A. I. (2006). Fisiología articular. Editorial médica Panamerica, Madrid, España.

Chapter 14
Robot Control and Tracking

14.1 Introduction

This chapter presents the computing of the dynamic model, the generation of trajectories using quadratic programming with geometric constraints, and nonlinear control for robot manipulators using the geometric algebra framework. The computing of the dynamics is carried out using screw theory with the iterative Newton–Euler algorithm in the motor algebra framework $G^{+}_{3,0,1}$.

14.2 Optimization Using Geometric Constraints

Figure 14.1 shows a general representation for Whole Body Torque Control (WBTC) for robots.

As Fig. 14.1 shows, the supervised planning of robot maneuvering is carried out using Quadratic Programming (QP) subject to constraints considering: (i) desired robot motion; (ii) obstacles in the workspace; (iii) robot kinematics. The WBTC outputs the references through time for the robot joints. The kinematics and the geometric constraints are formulated in conformal geometric algebra $G_{4,1}$, whereas the dynamic is computed using the Newton–Euler iterative algorithm in the motor algebra framework $G^{+}_{3,0,1}$.

14.2.1 Desired Motion

In the proposed framework, the motion planning is used to specify desired motions of the robot. One defines a motion task as an equation which is linear in the desired velocity vector of the robot joint $\dot{q}$. In general, a i-motion task is described in terms of the robot joint velocity

© Springer Nature Switzerland AG 2020

E. Bayro-Corrochano, *Geometric Algebra Applications Vol. II*,
https://doi.org/10.1007/978-3-030-34978-3_14

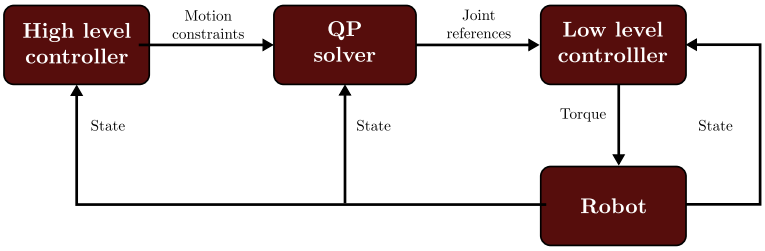


Fig. 14.1 Whole body torque control

$$A_i \dot{q} = b_i. \tag{14.1}$$

There are two kinds of motion tasks: (i) tasks in the space of the joints (configuration space) and (ii) tasks in the workspace.

For the space of the joints, the matrix $J_i \in \mathbb{R}^{n_j \times n}$ is used as selection matrix, where n_j is the DOF of the j-th joint and $\dot{q}_{d,j}$ is its desired joint velocity.

$$A_i = \begin{bmatrix} 0 & \cdots & 0 & I_{n_j \times n_j} & 0 & \cdots & 0 \end{bmatrix}, \quad b_i = \dot{q}_{d,j}. \tag{14.2}$$

In order to express the tasks in the workspace, defining the desired *twist* T_k^d for the k-th robot link

$$T_i^d = \begin{pmatrix} \omega_k^d \\ v_k^d \end{pmatrix}, \tag{14.3}$$

so one can write the task as

$$A_i = J_k, \tag{14.4}$$

$$b_i = T_k^d, \tag{14.5}$$

where $J_i \in \mathbb{R}^{6 \times n}$ is the Jacobian corresponding to a k-th link. When some components of the motions have to be restricted, each of these basic tasks of motions can be premultiplied with a selection matrix S, i.e., $SA_i\dot{q} = Sb_i$. For example, if the velocity of one link is restricted, the selection matrix is $S = [0_{3\times3} \quad I_{3\times3}]$.

The motion tasks for tracking in the configuration space or the workspace are formulated in terms of the joint velocities or space velocities included in b_i and they are computed using kinematic control, namely,

$$b_i = T_k^d + K_c(X_k^d - X_k), \tag{14.6}$$

where X_k and X_k^d are the real and desired poses of the k-th link and K_c is a positive control gain which guarantees asymptotic convergence.

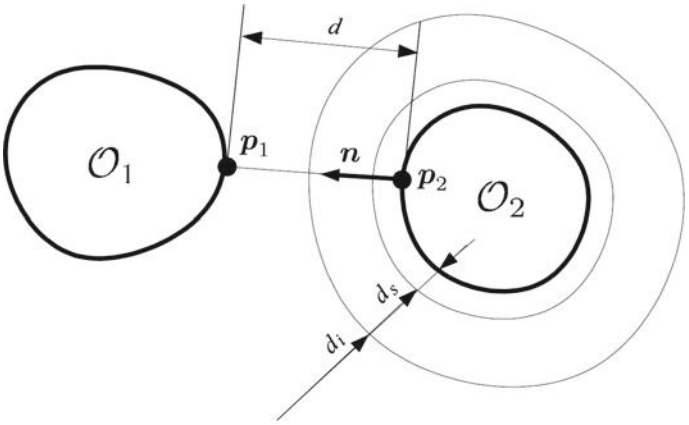


Fig. 14.2 Scheme for collision avoidance

14.2.2 Collision Avoidance

Collision avoidance is formulated as an inequality restriction in quadratic programming [1] as follows, consider one mobile convex object $\mathcal{O}_1$ and another static object $\mathcal{O}_2$ where p_1 y p_2 are the closest points and d is distance between them. Since $\mathcal{O}_1$ and $\mathcal{O}_2$ are convex objects, p_1 y p_2 move continuously in the limits of limits of $\mathcal{O}_1$ and $\mathcal{O}_2$ and d is continuously differentiable, see Fig. 14.2.

If d is lower than a threshold called the influence distance and denoted by d_i, the restriction for the velocity $\dot{d}$ can be defined as follows:

$$\dot{d} \geq -\epsilon \frac{d - d_s}{d_i - d_s}, \tag{14.7}$$

where ϵ is a positive coefficient to adjust the velocity of convergence and d_s, called the security distance, is a positive value $0 < d_s < d_i$.

The inequality (14.7) is known as the damping velocity and it tells that d should diminish very fast when d_i is smaller than d_i. As a result, d will never be smaller than d_s. The region where d is smaller than d_i is called the influence zone. Note that $\dot{d}$ has to satisfy the inequality (14.7) when p_1 protrudes the influence zone. If it doesn't, then $\dot{d}$ is re-stringed discontinuously. Thus, the value of ϵ should be adjusted depending upon the application.

$\dot{d}$ is computed as follows:

$$\dot{d} = n \cdot \dot{p}_1, \tag{14.8}$$

which means the inner product between $\dot{p}_1$, the point velocity, and n, the unit vector of the motion direction $n = (p_1 - p_2)/d$. The velocity of p_1 can be expressed as

$$\dot{p}_1 = J_{p_1}(q)\dot{q}, \tag{14.9}$$

where $J_{p_1}(q)$ is the Jacobian of $\mathcal{O}_1$ in the point p_1. Henceforth, the inequality (14.7) turns to be lineal w.r.t. $\dot{q}$

$$(n^T J_{p_1}(q))\dot{q} \geq -\epsilon \frac{d - d_s}{d_i - ds}. \tag{14.10}$$

The computation of a pair of close points between two colliding objects is not computationally easy. In this regard, we will use instead spheres of the conformal geometric algebra to delimit both links and the obstacles. This kind of representation helps greatly to formulate the geometric inequalities for quadratic programming in order to avoid collisions between links and obstacles. Thus, one can rewrite the inequality of Eq. (14.10) in terms of conformal geometric algebra as follows:

$$-(n \cdot J_{p_1}(q))\dot{q} \leq \epsilon \frac{d - d_s}{d_i - d_s}, \tag{14.11}$$

where n is computed as

$$n = \frac{((s_1 - s_2) \wedge E)E}{\|((s_1 - s_2) \wedge E)E\|}, \tag{14.12}$$

where $E = e_\infty \wedge e_0$ is the Minkowski plane. To compute the distance d, one uses following equation:

$$d = \sqrt{-2(s_1 \cdot s_2 - \tfrac{1}{2}\rho_1 - \rho_2)} - \rho 1 - \rho_2, \tag{14.13}$$

where ρ_i is the ratio of the sphere s_i. The Jacobian $J_{p_1}(q)$ can be computed using the sphere centers.

One can extend the idea for avoiding collisions, either between links and obstacles or between links, for that one covers each link with a sphere. Let the $s_{o,i}$ spheres delimiting the i-th obstacle and let $s_{l,j},\ j = 1, \ldots, n_l$, the n_l spheres delimiting the l-th link. The key idea is to find the minimum distance between the i-th obstacle and all the n_l spheres which delimit the l-th link

$$d_{k,i} = \min(d(s_{o,i}, s_{l,j})), \tag{14.14}$$

for each i-th obstacle, where $d(\cdot, \cdot)$ denotes the distance between the spheres and computed using Eq. (14.13). After computing the minimum distance $d_{k,i}$ which corresponds to the distance between the k-th sphere of the l-th link $s_{l,k}$ and the i-th obstacle $s_{o,i}$, one continues to compute the normal vector n_k according to Eq. (14.12) and the Jacobian $J_{l,k}$ of the sphere $s_{l,k}$ computed as mentioned before.

After these computations, one defines the inequality of Eq. (14.11) rewritten as follows:

$$E_i \dot{q} \leq f_i, \tag{14.15}$$

with

$$E_i = -(n_k \cdot J_{l,k}(q)), \tag{14.16}$$

$$f_i = \epsilon \frac{d_{k,i} - d_s}{d_i - d_s}. \tag{14.17}$$

Joint limits. A compact and efficient form to handle the velocity and its limits at the joints was originally proposed by Kanehiro et al. [1]. The restrictions for the velocity and for the limits of the joints are formulated as follows:

$$\hat{\dot{q}}^- \leq \dot{q} \leq \hat{\dot{q}}^+, \tag{14.18}$$

where $\hat{\dot{q}}^-$ and $\hat{\dot{q}}^+$ are the generalized limits of the joint velocities defined as

$$\hat{\dot{q}}_j^+ = \begin{cases} \dot{q}_j^+ \frac{(q_j^+ - q_j) - \varrho_s}{(\varrho_i - \varrho_s)} & \text{is } q_j^+ - q_j \leq \varrho_i \\ \dot{q}_j^+ & \text{other case,} \end{cases} \tag{14.19}$$

$$\hat{\dot{q}}_j^- = \begin{cases} \dot{q}_j^- \frac{(q_j - q_j^-) - \varrho_s}{(\varrho_i - \varrho_s)} & \text{is } q_j - q_j^- \leq \varrho_i \\ \dot{q}_j^+ & \text{other case,} \end{cases} \tag{14.20}$$

where $\hat{\dot{q}}_j^+, \hat{\dot{q}}_j^-$ is the j-th element of the vector $\hat{\dot{q}}^+$, $\hat{\dot{q}}^-$, q_j is the value of the j-th joint, q_j^+ y q_j^- are the upper and lower limits of the j-th joint and $\dot{q}_j^+$ and $\dot{q}_j^-$ are the upper and lower limits of the velocities of the j-th joint, where ϱ_i and ϱ_s are positive constants defined by the user and ϱ_i generally is called the influence distance.

14.2.3 Quadratic Programming

The formulation of the quadratic programming uses the restrictions described in Sects. 14.2.1 and 14.2.2. The equality restrictions involve the task of desired motions. The tasks of individual motions are concatenated in one matrix equation

$$A\dot{q} = b, \tag{14.21}$$

where $A = [A_1^T \quad A_2^T \quad \cdots \quad A_k^T]^T$ and $b = [b_1^T \quad b_2^T \quad \cdots \quad b_k^T]^T$.

Similarly for the inequality restrictions one uses Eq. (14.15) for each pair of closest points between the spheres defined and concatenated in the matrix E and the vector f. At last, one includes the inequalities for the limits of positions and angular velocities according to Eqs. (14.19) and (14.20).

Based on the previous comments and equations, the problem using quadratic programming becomes

$$\begin{aligned} \min_{\dot{q}} \quad & \mathcal{J} = \frac{1}{2}\dot{q}^T Q\dot{q} \\ \text{subject to} \quad & A\dot{q} = b \qquad (14.22) \\ & E\dot{q} \leq f \\ & \hat{\dot{q}}^- \leq \dot{q} \leq \hat{\dot{q}}^+. \qquad (14.23) \end{aligned}$$

The solution of the given probe of PQ will be denoted as $\dot{q}^*$ which represent the joint velocities fulfilling the given restrictions. To find a solution, we use the libraries of public domain.

14.2.4 Control

The inverse dynamics consists of finding the necessary torques for a certain acceleration of a rigid body. The inverse dynamics is used for the motion control of a system and to generate the trajectories. The computation of the dynamics can be formulated as

$$\tau = \text{ID}(model, \theta, \dot{\theta}, \ddot{\theta}), \qquad (14.24)$$

where $\theta, \dot{\theta}, \ddot{\theta}$, and τ are the variables of position, velocity, acceleration, and the forces/torques at the joints, respectively. The *model* represents the body characteristics as mass, distances, configuration, etc., which define completely a certain rigid body. In order to compute the dynamics of a rigid body, the recursive algorithm of Newton–Euler is used, which is explained in Sect. 11.4.

The motion equation for a robotic system is given by

$$\mathcal{M}(q)\ddot{q} + C(q, \dot{q})\dot{q} + G(q) = \tau. \qquad (14.25)$$

Using Eq. (14.25), the model in state variable space for the position and velocity is defined as

$$\dot{x}_1 = x_2, \qquad (14.26)$$

$$\dot{x}_2 = f(x_1, x_2) + B(x_1, x_2)u, \qquad (14.27)$$

$$y = x_1, \qquad (14.28)$$

where $x_1 = [q_1\, q_2, \ldots, q_n]^T$ y $x_2 = [\dot{q}_1\, \dot{q}_2, \ldots, \dot{q}_n]^T$ are the state variables for position and velocity, respectively, $u = \tau$ correspond to the torque values of the control signal, and $f(x_1, x_2) = -\mathcal{M}^{-1}(x_1)\,(C(x_1, x_2)x_2 + G(x_1))$, $B(x_1, x_2) = \mathcal{M}^{-1}(x_1)$, $y = x_1$ is the system output. For a 2 DOF, the dynamic model is computed using the

recursive Newton–Euler algorithm of Sect. 11.4. For a DOF robot manipulator, the resulting matrices M, V, G are given by Eqs. (11.141)–(11.142). Having obtained the model in the variable state space, one can then design the nonlinear controller.

Controller in terms of the inverse dynamics. The law control is computed using the inverse dynamics [2]. The controller has to follow certain trajectories, which are computed according to the control scheme using quadratic programming. The control law is given by

$$u = \mathcal{M}(q)a_d + C(q, \dot{q})\dot{q} + G(q), \tag{14.29}$$

where a_d stands for the desired dynamics given in Eq. (14.31). If the error is defined as $e_1 = x_d - x_1$, one can propose the model of the error equations as

$$\begin{aligned} \dot{e}_1 &= \dot{x}_d - x_2 = e_2, \\ \dot{e}_2 &= \ddot{x}_d - f(x_1, x_2) - B(x_1, x_2)u. \end{aligned} \tag{14.30}$$

Substituting the control law (14.29) in the second equation of the error model (14.30), one gets

$$\begin{aligned} \dot{e}_2 &= \ddot{x}_d - f(x_1, x_2) + B(x_1, x_2)u, \\ &= \ddot{x}_d - \mathcal{M}^{-1}(x_1)\left(C(x_1, x_2)x_2 + G(x_1)\right) + \mathcal{M}^{-1}(x_1)u, \\ &= \ddot{x}_d - \mathcal{M}^{-1}(x_1)\left(u - C(x_1, x_2)x_2 - G(x_1)\right), \\ &= \ddot{x}_d - a_d, \end{aligned}$$

this cancels the intern dynamic and it remains only $\ddot{x}_d$ y a_d. In order to make sure that the system will follow the trajectories, we propose a_d as follows:

$$\begin{aligned} a_d &= \ddot{x}_d - K_p\left(x_d - x_1\right) - K_d\left(\dot{x}_d - x_2\right), \\ &= \ddot{x}_d - K_p e_1 - K_d e_2, \end{aligned} \tag{14.31}$$

where $x_d = q^*$, $\dot{x}_d = \dot{q}^*$, and $\ddot{x}_d = \ddot{q}^*$ stand for the position reference, velocity, and acceleration, respectively, and corresponding to the joints obtained when the quadratic programming problem is solved. K_p and K_d are the $n \times n$ matrices. Thus, the errors are fed back with the error signal as follows:

$$\dot{e}_1 = x_2, \tag{14.32}$$

$$\dot{e}_2 = K_p e_1 + K_d e_2 o. \tag{14.33}$$

A simple option for the gain matrices K_p y K_d is given by

$$\begin{aligned} K_p &= \text{diag}\left(\omega_1^2,\ \omega_2^2,\ \ldots,\ \omega_n^2\right), \\ K_d &= \text{diag}\left(2\omega_1,\ 2\omega_2,\ \ldots,\ 2\omega_n\right). \end{aligned} \tag{14.34}$$

Resulting in a decoupled closed-loop system, where the response at each joint is equal to the response of a critically damped second-order system with natural frequency ω_i which determines the response velocity of the joint, or equivalently the rate of decay of the error during the tracking. Note that to compute u, one uses the inverse dynamics as follows:

$$u = \text{ID}(modelo, q, \dot{q}, a_d), \tag{14.35}$$

with a_d as given by Eq. (14.31).

14.3 Integral Sliding Mode Control

In the practice, the nominal robot model suffers from parametric variations, so one doesn't know exactly all the parameters which define the plant, and thus for a robust control, we resort to the integral sliding mode controller [3]. The design of the integral sliding mode presupposes the existence of a previously designed controller for the nominal plant without perturbations. If one adds to this controller a discontinuous term to force the occurrence of the sliding mode controller, one gets an output similar as if the plant was nominal with its controller in closed-loop despite the presence of external perturbations and parametric variations of the system.

We start considering as model

$$\dot{x}_1 = x_2, \tag{14.36}$$
$$\dot{x}_2 = f(x_1, x_2) + B(x_1, x_2)u + h(x_1, x_2, t), \tag{14.37}$$

where $h(x_1, x_2, t)$ comprises the perturbations attributable to parametric variations, non-modeled dynamics, and external perturbations. One assumes that it fulfills with the conditions *match* [3]. Moreover, $h(x_1, x_2, t)$ have to be bounded for scalar functions which are positive scalar functions known in the following form:

$$\|h(x_1, x_2, t)\| \leq h_i^+(x_1, x_2, t),\ i = 1, \ldots, n.$$

Henceforth, one can define the control as

$$u = u_0 + u_1, \tag{14.38}$$

where u_0 is the control for the nominal system and u_1 is the control signal for sliding modes which will be designed to reject the perturbation. Henceforth, the system error (14.30) considering the proposed control law and the perturbation can be written as follows:

$$\begin{aligned} \dot{e}_1 &= e_2, \\ \dot{e}_2 &= \ddot{x}_d - f(x_1, x_2) - B(x_1, x_2)u_0 - B(x_1, x_2)u_1 - h(x_1, x_2, t). \end{aligned} \tag{14.39}$$

We define the following sliding mode surface:

$$s = \sigma(x) + z, \tag{14.40}$$

where $\sigma(x)$ is a function of state variables of the system which is defined according to certain criterion and z is the integral term to be determined later on. Using $\sigma(x) = Ce_1 + e_2$. Substituting in the derivative of (14.40)Eq. 14.39, one gets

$$\dot{s} = Ce_2 + \ddot{x}_d - f(x_1, x_2) - B(x_1, x_2)u_0 - B(x_1, x_2)u_1 - h(x_1, x_2, t) + \dot{z}, \tag{14.41}$$

and defining

$$\dot{z} = Ce_2 + f(x_1, x_2) + B(x_1, x_2)u_0 - \ddot{x}_d,\ z(0) = -Ce_1(0) - e_2(0), \tag{14.42}$$

Eq. (14.41) becomes

$$\dot{s} = -B(x_1, x_2)u_1 - h(x_1, x_2, t). \tag{14.43}$$

Next the control u_1 is proposed. According to the theory of integral sliding modes, this control should be formulated as an algorithm of classic sliding modes of first order which rejects perturbations,

$$u_1 = K(x)\text{sign}\,(s)\,, \tag{14.44}$$

where $K(x)$ is chosen so that one forces the occurrence of sliding modes by the discontinuous control u_1; furthermore, they should occur from the beginning as $s(0) = \sigma(x(0)) + z(0) = 0$ and thus $\dot{s} = s = 0$. However, in this work, the algorithm of *super-twisting* is used for the computation of u_1

$$\begin{aligned} u_1 &= \alpha_1 |s|^{1/2} \text{sign}\,(s) + v, \\ \dot{v} &= \alpha_2 \text{sign}\,(s)\,. \end{aligned} \tag{14.45}$$

Substituting Eq. (14.45) in (14.43), the dynamics of the sliding surface is given by

$$\dot{s} = -B(x_1, x_2)(\alpha_1 |s|^{1/2} \text{sign}\,(s) + v) + h(x_1, x_2, t), \tag{14.46}$$

$$\dot{v} = \alpha_2 \text{sign}\,(s)\,. \tag{14.47}$$

According to the formulation of [4] with certain conditions for the gains α_1 and α_2, it is possible to show that the dynamics of the sliding mode surface is globally asymptotically stable. Regarding the previous concepts, one guarantees that the control u_1 rejects the perturbation $h(x_1, x_2, t)$, i.e., $B(x_1, x_2)u_1 = -h(x_1, x_2, t)$ and that the system errors are given as follows:

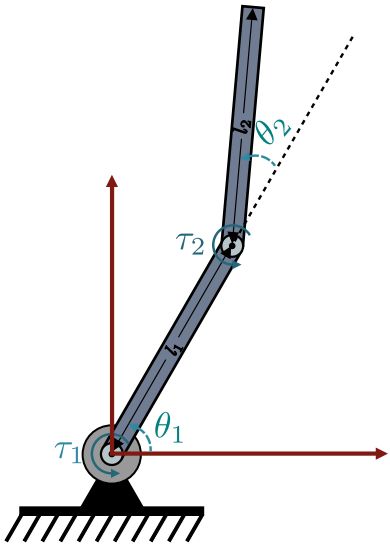


Fig. 14.3 Robot arm of 2 DOF

$$\begin{aligned}\dot{e}_1 &= e_2,\\ \dot{e}_2 &= \ddot{x}_d - f(x_1, x_2) - B(x_1, x_2)u_0,\end{aligned} \tag{14.48}$$

which is equivalent to the nominal system with the control signal u_0, which was our prime objective. To complete the control design, it is still needed to define the control u_0. Since (14.48) is equal to the presented in Eq. (14.30), u_0 is chosen as Eq. (14.38) of previous section.

14.4 Experimental Analysis

This section presents the dynamic modeling using the iterative Newton–Euler algorithm which is computed in the motor algebra $G^{+}_{3,0,1}$, the generation of trajectories using quadratic programming subject to geometric constraints developed using conformal geometric algebra $G_{4,1}$ and the application of an integral sliding mode controller with perturbations.

14.4.1 Parameters of the Dynamic Model

We use a serial robot of 2 DOF as shown in Fig. 14.3.

The robot specifications are link lengths $l_1 = l_2 = 1$ m and their masses $m_1 = m_2 = 1$ kg. The task is that the end-effector should follow a circular path in the plane

according to $[x_{ref},\ y_{ref}]^T = [0.25\cos(t) + 0.6,\ 0.25\sin(t) - 0.25]^T$. The angular positions are delimited in a range of $-3/2\pi \leq q_i \leq 3/2\pi$, as well the angular velocities in a range of $-3/2\pi$ rad/s $\leq \dot{q}_i \leq 3/2\pi$ rad/s. There are three obstacles delimited by circles of radius $\rho_{o,i} = 0.1$ m, which in turn describe circular movements around a spatial reference described before. To avoid collision between them as well as with the obstacles, the links are the limits each one by a set of cycles of radius $\rho_{l,i} = 0.05$ m. Moreover, it is used as an influence distance of $d_i = 0.25$ m.

14.4.2 Integral Sliding Mode Controller

In order to test the efficiency of the control using an integral sliding mode controller, in the simulation, the following perturbation was added:

$$\delta = [3\sin(4t),\ -\cos(3t)]^T\ ,\ m_1 = m_2 = 1.1\,\text{kg}.$$

The parameters for the PD controller for the inverse dynamic are

$$K_p = \text{diag}\,(9, 9)\,,\quad K_d = \text{diag}\,(6, 6)\,,\quad C = \text{diag}\,(10, 10)\,,$$
$$\alpha_1 = \text{diag}\,(13, 5)\,,\ \alpha_2 = \text{diag}\,(1.5, 0.8)\,.$$

Figure 14.4 shows the tracking of the reference in the tracking plane, whereas Fig. 14.5 presents the tracking error in each axis of the plane. One notices that between the first two seconds the tracking error in the space goes asymptotically to 0. At $t \approx 12.5$, the error is transient due to the change of trajectory to avoid a collision with an object, see Fig. 14.8, and thereafter the tracking returns immediately to its correct path.

In Fig. 14.6, one observes the angular positions obtained compared with the references obtained by the trajectory generator, which are located at the limits of the imposed angular position restrictions.

Figure 14.7 describes the obtained control signals. Note that these signals have a small oscillatory behavior, which is due to the perturbation rejection by the controller and the parametric variations as well. Moreover, high-frequency components are also caused when the controller avoids the object collision. It is worthy to mention that despite the uncertainties in the simulation, the control signals present a lower amplitude b.

Finally, Fig. 14.8 presents the sequence of the robot behavior each second, where one can see that the end-effector avoids the collision with the obstacles satisfactorily while it fulfills with the expected trajectory.

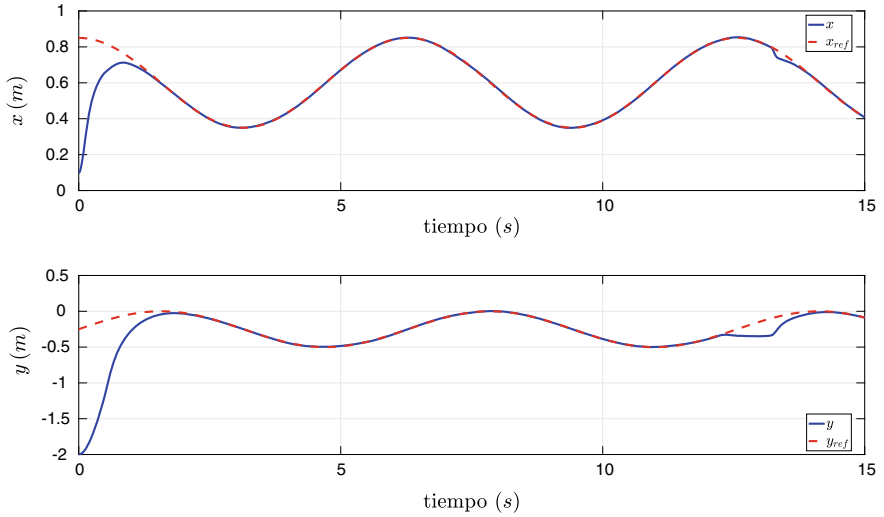


Fig. 14.4 Following the reference in the space

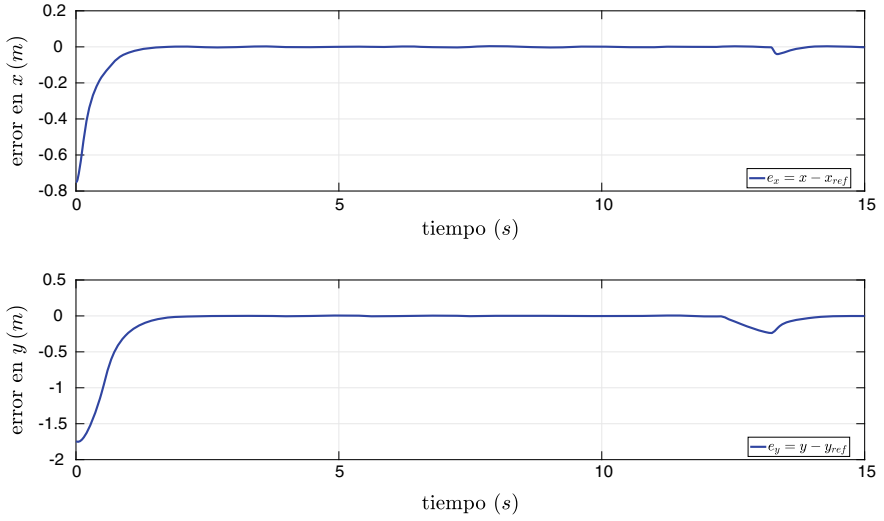


Fig. 14.5 Error of tracking w.r.t the reference in the space

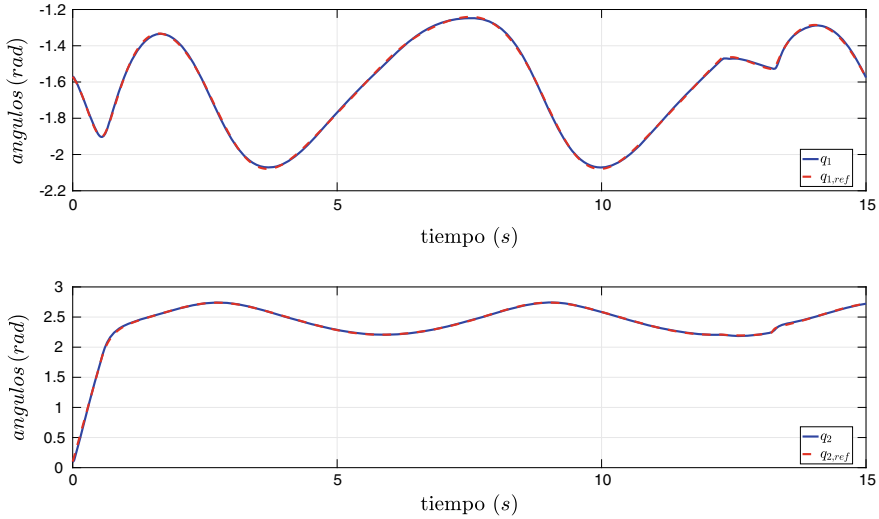


Fig. 14.6 Angular position for each joint

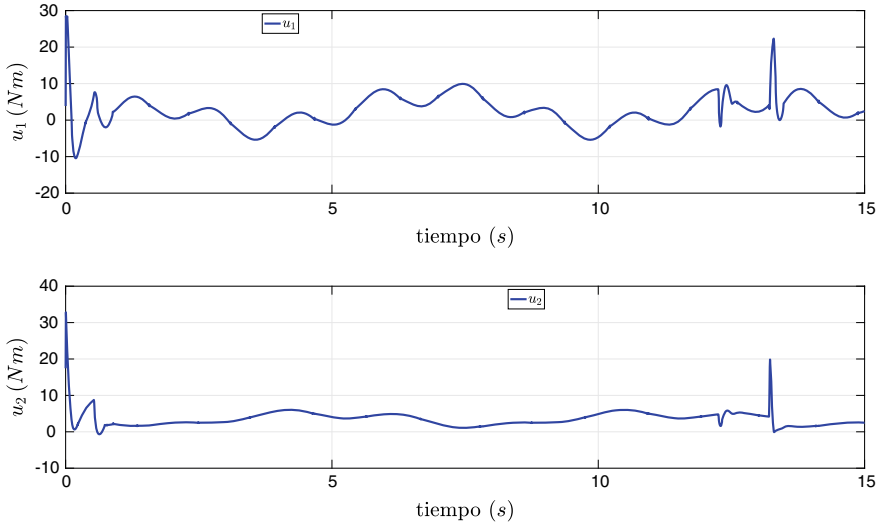


Fig. 14.7 Seales de control

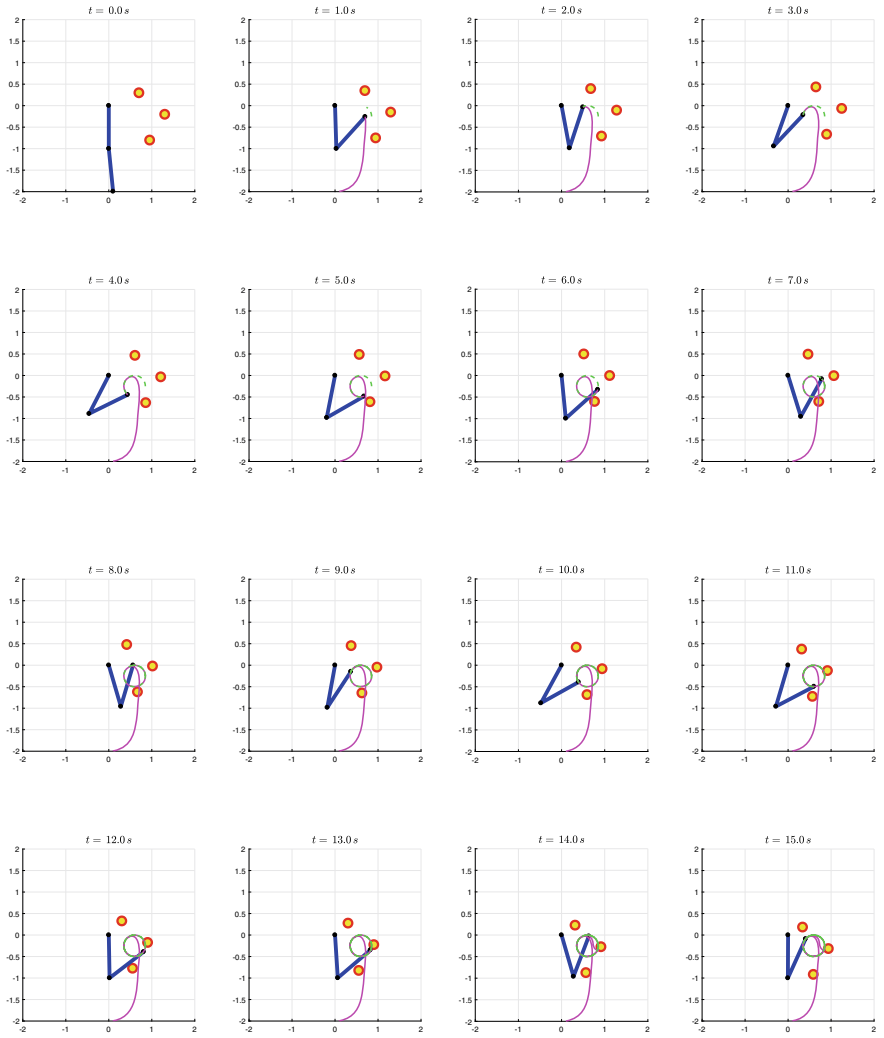


Fig. 14.8 Sequence of behavior of the 2-DOF robot arm

14.5 Conclusion

This chapter presents the computing of the dynamic model, the generation of trajectories using quadratic programming with geometric constraints, and nonlinear control for robot manipulators using the geometric algebra framework. The computation of the dynamics is carried out using screw theory with the iterative Newton–Euler algorithm in the motor algebra framework $G^{+}_{3,0,1}$. This algorithm is new as it uses screw lines and motors (rotors and translators) instead of vectors and

matrices. In order to generate the robot navigation trajectories, we resort to theory of optimization using quadratic programming with geometric constraints. Note that these constraints are designed using screw theory and spheres in the conformal geometric algebra framework $G_{4,1}$. Nonlinear controllers for collision avoidance are implemented using integral sliding modes involving external perturbations. The experimental analysis presents the maneuvering of a robot manipulator subject to external perturbations. Our results confirm the efficient performance of the nonlinear control for tracking avoiding obstacles even with external perturbations.

References

1. Kanehiro, F., Lamiraux, F., Kanoun, O., Yoshida, E., & Laumond, J.-P. (2008). A local collision avoidance method for non-strictly convex polyhedral. In *Robotics: Science and systems IV*, Zurich, Switzerland.
2. Spong, M. W., Hutchinson, S., & Vidyasagar, M. (2006). *Robot modeling and control*. New York: Wiley.
3. Utkin, V. I. (1992). *Sliding modes in control and optimization*. Berlin: Springer-Verlag.
4. Moreno, J. A., & Osorio, M. (2008). A Lyapunov approach to second-order sliding mode controllers and observers. In *2008 47th IEEE Conference on Decision and Control* (pp. 2856–2861).

Part IV
Applications I: Robot Vision, Quadrotor

Chapter 15
Rigid Motion Estimation Using Line Observations

15.1 Introduction

This chapter is dedicated to the estimation of 3D Euclidean transformation using motor algebra. Two illustrations of estimation procedures are given: the first uses a batch approach for the estimation of the unknown 3D transformation between the coordinate reference systems of a robot neck, or arm, and of a digital camera. This problem is called the hand–eye problem and it is solved using a motion-of-lines model. The second illustration uses a recursive estimation method based on Kalman filter techniques. After introducing the Motor Extended Kalman Filter (MEKF), we estimate 3D rigid motion using the MEKF and 3D lines gained by a visual robot system. These two approaches show that the task of estimating 3D rigid motion is easier using motor algebra because the motion-of-lines model is linear.

15.2 Batch Estimation Using SVD Techniques

In this section, we will illustrate the use of motor algebra for solving an exemplary task of visually guided robotics. We choose the so-called hand–eye calibration problem. This kind of task may be found in the area of visually guided robotics, where cameras are attached to robot arms or mounted on a vehicle and they have to be directed toward a goal. On the one hand, the cameras capture visual cues in the 3D visual space and employ their own world reference coordinate system. On the other hand, the robot arm or vehicle moves relative to a reference coordinate system. If we compute the intrinsic and extrinsic parameters through the camera's movements, we can find the geometrical relationship between the camera position and the world coordinates. The position of the robot arm or vehicle, on the other hand, is always known, owing to the angular position of the step motors of the device, which are permanently controlled by the computer. The problem of hand–eye calibration arises when we try to determine

© Springer Nature Switzerland AG 2020

E. Bayro-Corrochano, *Geometric Algebra Applications Vol. II*,
https://doi.org/10.1007/978-3-030-34978-3_15

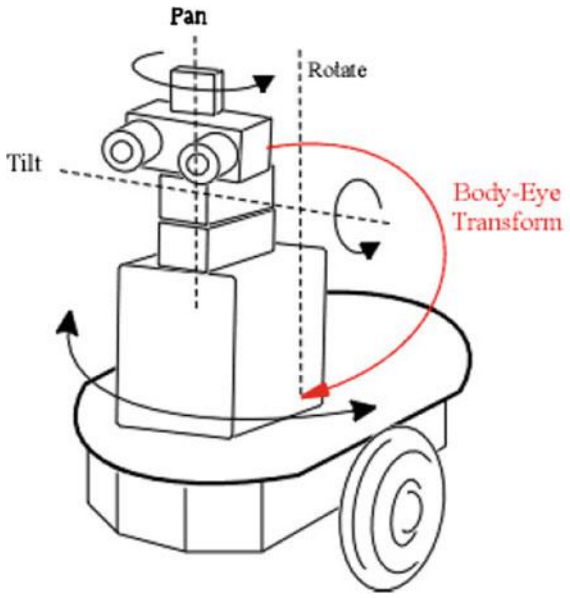


Fig. 15.1 Abstraction of the hand–eye system

the group transformation between the reference coordinates of the mechanical device and the coordinate frame of the camera.

An abstraction of the geometry of a camera mounted on a robot arm is depicted in Fig. 15.1. The classical way to describe the hand–eye problem in mathematical terms is by using transformation matrices. This problem was originally formulated by Shiu and Ahmad [1] and Tsai and Lenz [2] as a matrix equation of a Euclidean transformation:

$$\mathrm{AX} = \mathrm{XB}, \tag{15.1}$$

where the matrices $\mathrm{A} = \mathrm{A}_1\mathrm{A}_2^{-1}$ and $\mathrm{B} = \mathrm{B}_1\mathrm{B}_2^{-1}$ express the elimination of the transformation between hand–base and world. From Eq. (15.1), the following matrix equation and a vector equation can be derived by splitting the Euclidean transformation into rotation and translation components:

$$\mathrm{R}_A\mathrm{R}_X = \mathrm{R}_X\mathrm{R}_B \tag{15.2}$$

$$(\mathrm{R}_A - \mathrm{I})\mathrm{t}_X = \mathrm{R}_X\mathrm{t}_B - \mathrm{t}_A. \tag{15.3}$$

In the literature, we find a variety of methods to estimate the rotation matrix R_X from Eq. (15.2) (see the survey by Wang [3]), most of which first estimate the rotation matrix decoupled from the translation component. Tsai and Lenz showed that to solve the problem at least two motions are required, with rotations having

no parallel axes [2]. The most relevant approaches employ either the axis and the angle of rotation [1, 2], quaternions [4], dual quaternions [5], or a canonical matrix representation [6].

Horaud and Dornaika [7] were the first to apply a nonlinear method to compute R_X and t_X simultaneously. In their work, they showed the instability of the computation of the matrices A_i, given the projective matrices $\mathrm{M}_i = \mathrm{SA}_i = \mathrm{SR}_{\mathrm{A}_i}\mathrm{St}_{\mathrm{A}_i}$, where S is the matrix of the camera-intrinsic parameters. Let us assume that the matrix of the intrinsic parameters S remains constant during camera motion and that one extrinsic calibration A_2 is known. By introducing $\mathrm{N}_i = \mathrm{SR}_{\mathrm{A}_i}$ and $\mathrm{n}_i = \mathrm{St}_{\mathrm{A}_i}$ and replacing $\mathrm{X} = \mathrm{A}_2\mathrm{Y}$, we get as the hand–eye unknown Y. Thus, Eq. (15.1) can be reformulated as

$$\mathrm{A}_2^{-1}\mathrm{A}_1\mathrm{Y} = \mathrm{YB}. \tag{15.4}$$

Now, if $\mathrm{A}_2^{-1}\mathrm{A}_1$ is written as a function of the projection parameters, it is possible to obtain an expression fully independent of the intrinsic parameters S, that is,

$$\mathrm{A}_2^{-1}\mathrm{A}_1 = \begin{bmatrix} \mathrm{N}_2^{-1}\mathrm{N}_1 & \mathrm{N}_2^{-1}(\mathrm{n}_1 - \mathrm{n}_2) \\ 0_3^T & 1 \end{bmatrix} = \begin{bmatrix} \mathrm{R} & t \\ 0_3^T & 1 \end{bmatrix}. \tag{15.5}$$

Taking into consideration the selected matrices and relations, this result allows us to reconsider the formulation of the hand–eye problem employing the standard Eq. (15.1), which can then be solved using all the known methods, including the one presented in this chapter.

Another relevant contribution to the hand–eye problem was made by Chen [8]. Employing a geometric point of view, he formulated the hand–eye calibration problem using screw theory. Using this method, Chen discovered that the hand–eye transformation is fully independent of the pitch and the angle of the camera and hand motions. The unknown transformation simply depends upon the parameters which relate to the screw axis lines of the hand motion and of the camera motion. In this chapter, we will use a unitary motor that is completely isomorphic with the unit screw, and we will present an algorithm, different from the one used by Horaud and Dornaika [7], to compute rotation and translation simultaneously in a linear manner.

15.2.1 Solving AX = XB *Using Motor Algebra*

In terms of motors, system Eq. (15.1) can be expressed as

$$\boldsymbol{M}_A\boldsymbol{M}_X = \boldsymbol{M}_X\boldsymbol{M}_B \tag{15.6}$$

or as

$$\boldsymbol{M}_A = \boldsymbol{M}_X\boldsymbol{M}_B\tilde{\boldsymbol{M}}_X, \tag{15.7}$$

where $\boldsymbol{M}_A = \boldsymbol{A} + I\boldsymbol{A}'$, $\boldsymbol{M}_B = \boldsymbol{B} + I\boldsymbol{B}'$, and $\boldsymbol{M}_X = \boldsymbol{R} + I\boldsymbol{R}'$. Next, we simplify this equation to show the motor relation between the motor line axis of the camera $\boldsymbol{L}_A$ and the motor line axis of the hand $\boldsymbol{L}_B$.

We can isolate the scalar part of $\boldsymbol{M}_A$ by using the grade operator $\langle \boldsymbol{M}_A \rangle_0 \equiv \langle \boldsymbol{M}_A \rangle$, according to Eq. (4.63), and we can then use the previous equation to get

$$\begin{aligned}\langle \boldsymbol{M}_A \rangle &= \frac{1}{2}(\boldsymbol{M}_X \boldsymbol{M}_B \tilde{\boldsymbol{M}}_X + \boldsymbol{M}_X \tilde{\boldsymbol{M}}_B \tilde{\boldsymbol{M}}_X), \\ &= \frac{1}{2}\boldsymbol{M}_X(\boldsymbol{M}_B + \tilde{\boldsymbol{M}}_B)\tilde{\boldsymbol{M}}_X = \boldsymbol{M}_X \langle \boldsymbol{M}_B \rangle \tilde{\boldsymbol{M}}_X, \\ &= \boldsymbol{M}_X \tilde{\boldsymbol{M}}_X \langle \boldsymbol{M}_B \rangle = \langle \boldsymbol{M}_B \rangle. \end{aligned} \tag{15.8}$$

Thus, we are able to equalize the scalar parts of $\boldsymbol{M}_A$ and $\boldsymbol{M}_B$, and then, using Eq. (4.45), we obtain

$$\cos\left(\frac{\theta_A}{2} + I\frac{d_A}{2}\right) = \cos\left(\frac{\theta_B}{2} + I\frac{d_B}{2}\right), \tag{15.9}$$

which may also be written by separating the real and dual terms:

$$\begin{aligned}\cos\left(\frac{\theta_A}{2}\right) &= \cos\left(\frac{\theta_B}{2}\right), \\ d_A \sin\left(\frac{\theta_A}{2}\right) &= d_B \sin\left(\frac{\theta_B}{2}\right). \end{aligned} \tag{15.10}$$

In this way, we are sure that only the bivector terms of $\boldsymbol{M}_A$ and $\boldsymbol{M}_B$ will contribute to the computation of the unknown $\boldsymbol{M}_X$. Finally, using Eq. (4.45), the hand–eye equation reduces to

$$\begin{aligned}\sin\left(\frac{\theta_A}{2} + I\frac{d_A}{2}\right)\boldsymbol{L}_A &= \boldsymbol{M}_X\left(\sin\left(\frac{\theta_B}{2} + I\frac{d_B}{2}\right)\boldsymbol{L}_B\right)\tilde{\boldsymbol{M}}_X, \\ &= \sin\left(\frac{\theta_B}{2} + I\frac{d_B}{2}\right)\boldsymbol{M}_X \boldsymbol{L}_B \tilde{\boldsymbol{M}}_X. \end{aligned} \tag{15.11}$$

If θ_A and θ_B do not abandon the range from 0° to 360°, we can neglect the sines to get the simplified expression

$$\boldsymbol{L}_A = \boldsymbol{M}_X \boldsymbol{L}_B \tilde{\boldsymbol{M}}_X, \tag{15.12}$$

which shows that in this kind of problem formulation, the rotation and pitch of $\boldsymbol{M}_A$ and $\boldsymbol{M}_B$ are always equal throughout all the hand movements and therefore can be ignored in the computations. It will suffice to focus on only the rotation axes of the involved motors, that is, Eq. (15.12) is reduced to just the motion of the line axis of the hand $\boldsymbol{L}_B$ toward the line axis of the camera $\boldsymbol{L}_A$. This characteristic, known as the congruence theorem, has been pointed out by Chen [8]. However, thanks to the

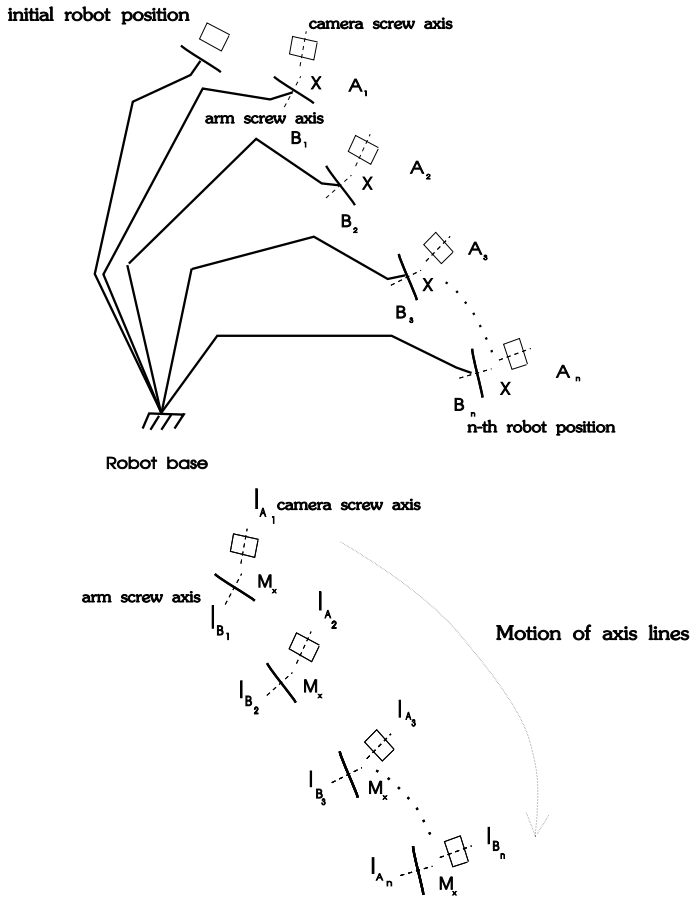


Fig. 15.2 The hand–eye system as the motion of related axes lines

use of motor algebra, the proof of this theorem has now been reduced to a single step, shown by Eq. (15.8). This simplification of the hand–eye problem is depicted in Fig. 15.2.

Since the hand–eye problem pertains to the motion of lines, Eq. (5.18) can be used to estimate the unknown 3D transformation which relates to the screw line axes of both the hand and the camera:

$$\boldsymbol{L}_A = \boldsymbol{a} + I\boldsymbol{a}' = \boldsymbol{R}\boldsymbol{b}\tilde{\boldsymbol{R}} + I(\boldsymbol{R}\boldsymbol{b}\tilde{\boldsymbol{R}}' + \boldsymbol{R}\boldsymbol{b}'\tilde{\boldsymbol{R}} + \boldsymbol{R}'\boldsymbol{b}\tilde{\boldsymbol{R}}), \tag{15.13}$$

where $\boldsymbol{a}$, $\boldsymbol{a}'$, $\boldsymbol{b}$, and $\boldsymbol{b}'$ are spanned by bivectors. Separating real and dual terms,

$$\begin{aligned} \boldsymbol{a} &= \boldsymbol{R}\boldsymbol{b}\tilde{\boldsymbol{R}} \\ \boldsymbol{a}' &= \boldsymbol{R}\boldsymbol{b}\tilde{\boldsymbol{R}}' + \boldsymbol{R}\boldsymbol{b}'\tilde{\boldsymbol{R}} + \boldsymbol{R}'\boldsymbol{b}\tilde{\boldsymbol{R}}, \end{aligned} \tag{15.14}$$

and multiplying from the right with motor $\boldsymbol{R}$, using the relation $\tilde{\boldsymbol{R}}\boldsymbol{R}' + \tilde{\boldsymbol{R}}'\boldsymbol{R} = 0$, we get the following multivector relationships:

$$\begin{aligned} \boldsymbol{a}\boldsymbol{R} - \boldsymbol{R}\boldsymbol{b} &= 0, \\ (\boldsymbol{a}'\boldsymbol{R} - \boldsymbol{R}\boldsymbol{b}') + (\boldsymbol{a}\boldsymbol{R}' - \boldsymbol{R}'\boldsymbol{b}) &= 0. \end{aligned} \tag{15.15}$$

These equations can be expressed in a matrix consisting of the scalar and bivector terms and the outer product as follows:

$$\begin{bmatrix} \boldsymbol{a} - \boldsymbol{b} & [\boldsymbol{a} + \boldsymbol{b}]_\times & \boldsymbol{0}_{3\times 1} & \boldsymbol{0}_{3\times 3} \\ \boldsymbol{a}' - \boldsymbol{b}' & [\boldsymbol{a}' + \boldsymbol{b}']_\times & \boldsymbol{a} - \boldsymbol{b} & [\boldsymbol{a} + \boldsymbol{b}]_\times \end{bmatrix} \begin{bmatrix} \boldsymbol{R} \\ \boldsymbol{R}' \end{bmatrix} = 0, \tag{15.16}$$

where the matrix—which we will call $\boldsymbol{D}$—is a 6×8 matrix, and the vector of unknowns $(\boldsymbol{R}, \boldsymbol{R}')^T$ is eight-dimensional. The notation $[\boldsymbol{a} + \boldsymbol{b}]_\times$ stands for the vector cross product as an antisymmetric matrix. Recall that we have two constraints on the unknowns, so that the result is a unit motor with the properties

$$\boldsymbol{R}\tilde{\boldsymbol{R}} = 1 \qquad \text{and} \qquad \boldsymbol{R}\boldsymbol{R}' = 0. \tag{15.17}$$

So far, we have six equations and two constraints. However, because the unit bivectors $\boldsymbol{a}$ and $\boldsymbol{b}$ are perpendicular to the bivectors $\boldsymbol{a}'$ and $\boldsymbol{b}'$, respectively, we conclude that two of the equations are necessarily redundant. That is not a surprise, really, because we already know that at least two lines are required to estimate 3D motion from their pertinent correspondences [9]. The result is that at least two motions of the hand–eye system are required in order to compute two lines from the involved screws. Chen [8] clearly noted this fact and analyzed the uniqueness of the problem. He proved geometrically that even in the case of two parallel rotation-axis lines, it is still possible to compute all the parameters up to the pitch.

15.2.2 Estimation of the Hand–Eye Motor Using SVD

We reduced the hand–eye problem to Eq. (15.16), which depends only on 3D bivectors. Now, for the estimation of the unknown rigid motion, we employ the Singular Value Decomposition (SVD) method [10], which is actually a vector approach for finding singular values. We are able to use this method since we are dealing only with bivectors. If we were dealing with a simultaneous estimation of multivectors of different grade, we would, of course, extend the SVD method to encompass a multivector concept.

Let us consider that $n \geq 2$ motions are available. Employing SVD, we build the following $6n \times 8$ matrix:

$$\boldsymbol{C} = \begin{bmatrix} \boldsymbol{D}_1^T & \boldsymbol{D}_2^T & \cdots & \boldsymbol{D}_n^T \end{bmatrix}^T. \tag{15.18}$$

Because 3D motion has six degrees of freedom, in the case of noise-free data, the greatest possible value of this matrix will be rank 6. Let us analyze this matrix in more detail. In the case of noise-free data, for which equations were calculated basically using geometric and algebraic concepts, we would expect that the null space necessarily contains at least the solution $(\boldsymbol{R}, \boldsymbol{R}')$. The solution $(\mathbf{0}_{4\times1}, \boldsymbol{R})$ ("pure rotation") is the trivial one, and thus we are able to reaffirm that the matrix indeed has a maximum rank of 6. In the particular case, where all the $\boldsymbol{b}$-axis lines are mutually parallel, one degree of freedom remains constant; in this case, the matrix will be of rank 5.

For the solution of Eq. (15.18), we use the SVD method. This procedure decomposes the matrix $\boldsymbol{C}$ into three matrices, as follows: $\boldsymbol{C} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$. The columns of the $\boldsymbol{U}$ and $\boldsymbol{V}$ matrices correspond to the left and right singular vectors, respectively, and $\boldsymbol{\Sigma}$ is a diagonal matrix with singular values. Since the rank of the matrix $\boldsymbol{C}$ is 6, the last two right singular vectors $\boldsymbol{v}_7$ and $\boldsymbol{v}_8$ correspond to the two vanishing singular values which span the null space of $\boldsymbol{C}$. For convenience, these will now be expressed in terms of two 4×1 vectors, $\boldsymbol{v}_7^T = (\boldsymbol{u}_1, \boldsymbol{v}_1)^T$ and $\boldsymbol{v}_8^T = (\boldsymbol{u}_2, \boldsymbol{v}_2)^T$. Since $(\boldsymbol{R}, \boldsymbol{R}')^T$ is a null vector of $\boldsymbol{C}$, specifically $\boldsymbol{C}(\boldsymbol{R}, \boldsymbol{R}')^T = 0$, then it must be expressed as a linear combination of $\boldsymbol{v}_7$ and $\boldsymbol{v}_8$, as follows:

$$\begin{bmatrix} \boldsymbol{R} \\ \boldsymbol{R}' \end{bmatrix} = \alpha \begin{bmatrix} \boldsymbol{u}_1 \\ \boldsymbol{v}_1 \end{bmatrix} + \beta \begin{bmatrix} \boldsymbol{u}_2 \\ \boldsymbol{v}_2 \end{bmatrix}.$$

Now, taking into account the two degrees of freedom imposed by Eq. (15.17), we obtain two quadratic equations in α and β:

$$\alpha^2 \boldsymbol{u}_1^T \boldsymbol{u}_1 + 2\alpha\beta \boldsymbol{u}_1^T \boldsymbol{u}_2 + \beta^2 \boldsymbol{u}_2^T \boldsymbol{u}_2 = 1, \tag{15.19}$$

$$\alpha^2 \boldsymbol{u}_1^T \boldsymbol{v}_1 + \alpha\beta(\boldsymbol{u}_1^T \boldsymbol{v}_2 + \boldsymbol{u}_2^T \boldsymbol{v}_1) + \beta^2 \boldsymbol{u}_2^T \boldsymbol{v}_2 = 0. \tag{15.20}$$

If we take into consideration that $\alpha \neq 0$, $\beta \neq 0$, and without loss of generality, we assume that $\boldsymbol{u}_1^T \boldsymbol{v}_1 \neq 0$, we can set $\mu = \alpha/\beta$ and substitute in Eq. (15.20) to obtain two solutions for μ. We then substitute the relation $\alpha = \mu\beta$ into Eq. (15.19) to obtain the following quadratic expression:

$$\beta^2(\mu^2 \boldsymbol{u}_1^T \boldsymbol{u}_1 + \mu(2\boldsymbol{u}_1^T \boldsymbol{u}_2) + \boldsymbol{u}_2^T \boldsymbol{u}_2) = 1, \tag{15.21}$$

which yields two solutions of opposite sign (this sign variation is simply an effect of sign invariance of the solution). Either term $\left[\boldsymbol{R}, \boldsymbol{R}'\right]^T$ or $\left[-\boldsymbol{R}, -\boldsymbol{R}'\right]^T$ will satisfy the motion equations and the involved constraints.

Since the equation also contains μ squared, we should also consider the viability of the other two solutions. Here, we can see that the second solution for μ causes the factor on the left-hand side of Eq. (15.21) to disappear. This corresponds to the solution $(\mathbf{0}_{4\times1}, \boldsymbol{R})$ and clearly does not satisfy the first constraint of Eq. (15.17). The algorithm, then, can be summarized as consisting of the following procedure:

1. Consider n hand motions $(\boldsymbol{b}_i, \boldsymbol{b}'_i)$ and their corresponding camera motions $(\boldsymbol{a}_i, \boldsymbol{a}'_i)$. Check to see if their scalar terms are equal (Chen's invariance theorem). By extracting the line directions and moments of the screw axes lines, construct the matrix $\boldsymbol{C}$ as in Eq. (15.18).
2. Apply the SVD procedure to $\boldsymbol{C}$ and check to see that two singular values are almost equal to zero. In the case of noisy data, keep the four largest singular values. Then select the related right singular vectors $\boldsymbol{v}_7$ and $\boldsymbol{v}_8$.
3. Set the coefficients for $\alpha^2 \boldsymbol{u}_1^T \boldsymbol{v}_1 + \alpha\beta(\boldsymbol{u}_1^T \boldsymbol{v}_2 + \boldsymbol{u}_2^T \boldsymbol{v}_1) + \beta^2 \boldsymbol{u}_2^T \boldsymbol{v}_2 = 0$, then solve, finding two solutions for μ.
4. Use these two values of μ to solve the equation $\beta^2(\mu^2 \boldsymbol{u}_1^T \boldsymbol{u}_1 + \mu(2\boldsymbol{u}_1^T \boldsymbol{u}_2) + \boldsymbol{u}_2^T \boldsymbol{u}_2) = 1$, and then select the solution with the largest value to compute α and then β.
5. The final solution will be $\alpha \boldsymbol{v}_7 + \beta \boldsymbol{v}_8$.

15.3 Experimental Results

In this section, we test this algorithm to compare its performance with a two-step algorithm similar to the one introduced by Chou and Kamel [4]. These authors estimated quaternion rotation $\boldsymbol{q}$ directly from the equation $\boldsymbol{aq} = \boldsymbol{qb}$ and then they computed the rotation matrix R_X and solved the translation t_X component using the vector Eq. (15.3).

The experiments were carried out using a computer simulation. First, n hand motions $(\mathrm{R}_b, \mathrm{t}_b)$ were created and Gaussian noise with a relative standard deviation of 1% was added so as to simulate the inaccuracy of angle readings. To simulate the hand–eye scenario, we generated camera motions $(\mathrm{R}_a, \mathrm{t}_a)$, similarly adding Gaussian noise of varying standard deviation. In this case, the noise was added as absolute value to the rotation-axis direction and as relative value to both the angle and the translation. In order to compute the estimated rotor $\boldsymbol{R}$ and the translation component $\boldsymbol{t}$ between the hand and camera, the algorithm was run 1,000 times for each value of added noise. The quantification of both algorithms was done according to the Root Mean Square (RMS) of the absolute errors in the rotation-unit rotor $\|\boldsymbol{R} - \hat{\boldsymbol{R}}\|$ and the RMS of the relative errors in the translation $\|\boldsymbol{t} - \hat{\boldsymbol{t}}\|/\|\boldsymbol{t}\|$.

For the first test, a set of 20 hand motions was prepared using different rotation axes and large rotation angles and a translation that varied from 10 to 20 mm. Figure 15.3 compares the results of our algorithm, labeled MOTOR, and those of the two-step algorithm, labeled SEPARATE.

The upper graph shows RMS rotation error and the lower graph shows RMS relative to translation error. Our algorithm substantially outperforms the two-step algorithm. By computing the rotation simultaneously with the translation, we were able to get a much better estimation of the rotation than was the case using a separate computation.

In the second test, we wanted to explore the estimation performance of both algorithms using zero translation. As expected, in this context, the behavior of both

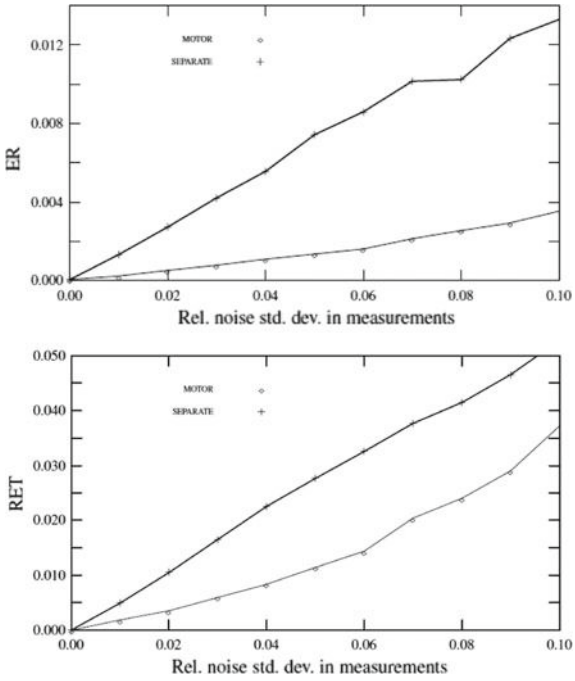


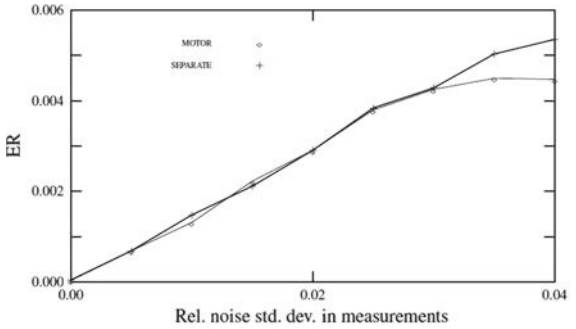
Fig. 15.3 Behavior of the proposed algorithm (MOTOR) and of a two-step algorithm (SEPARATE) with noise variation. ER stands for error in rotation and RET for relative error in translation

algorithms is almost the same (see Fig. 15.4). This effect is easy to explain if we consider Eq. (15.16). Since the translation is zero, the dual parts of the measurements $(\boldsymbol{a}', \boldsymbol{b}')$ become zero. Therefore, the left lower block of the matrix in Eq. (15.16) will disappear, which obliges the separate computation of $\boldsymbol{R}$ and $\boldsymbol{R}'$.

In the last experiment, we were interested in the performance of both algorithms when the noise level is kept constant and the number of motions is gradually increased. Generally speaking, one would expect much better estimation with the use of a greater number of hand and camera motions. The noise level was kept at 5% and the number of motions varied from 2 to 20. Figure 15.5 shows that from the fourth motion onward our algorithm had a superior performance.

15.4 Discussion

This chapter proposes the Clifford, or geometric, algebra for computations in visually guided robotics. In looking for other suitable ways of representing the motion of geometric primitives, it turns out that the algebra of motors is well suited to express 3D kinematics, since its use allows us to linearize the nonlinear 3D rigid motion transformation.



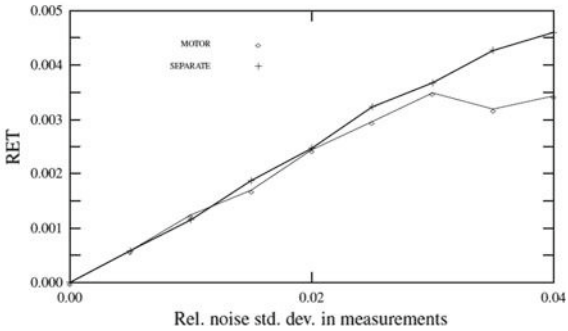
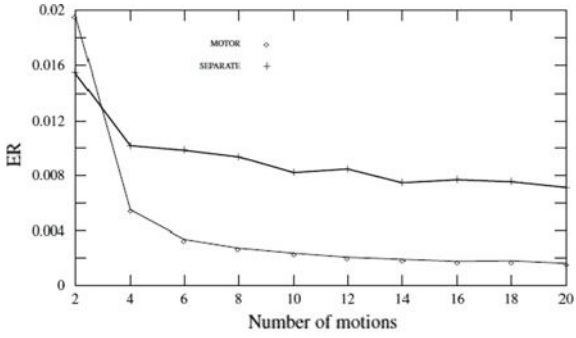


Fig. 15.4 Performance of both algorithms in absence of translation



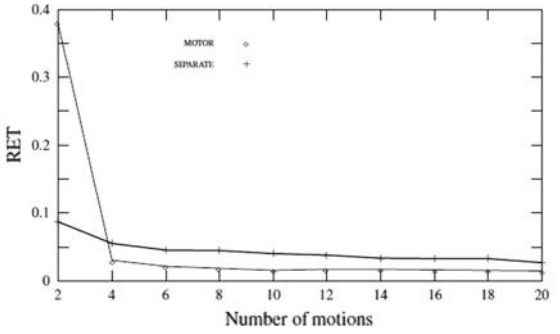


Fig. 15.5 Errors in rotation (upper) and translation (below) as a function of the number of hand and camera motions

In the literature, it has been shown that the invariance of the angle and the pitch of the screws of the camera and hand help to reduce the complexity of the hand–eye calibration problem. We used this fundamental idea to simplify the hand–eye problem to a problem of motion of lines. For this case, we used the algebra of motors, which is indicated for problems involving the algebra of lines. The resultant simplified parameterization of the problem enabled us to establish a linear homogeneous system for calculating the motor parameters. By computing the null space using SVD and considering a few constraints of dual rotors, we were able to devise a simple algorithm, one that obviates the need for nonlinear computations. In this work, it can be seen that the algebraic structure of the resultant equations helps us to better understand the performance of the algorithm.

The next section of this chapter is devoted to the application of Kalman filter techniques for the estimation of 3D rigid motion using 3D line observations. First, we will outline Kalman filter techniques and describe in detail the motor extended Kalman filter, and then we will apply these techniques for 3D rigid motion estimation using observed lines captured by a robotic visual system.

15.5 Recursive Estimation Using Kalman Filter Techniques

The Kalman filter is a linear recursive algorithm that is unbiased and of minimum variance. It is employed to estimate optimally the unknown state of a linear dynamic system using noisy data which are taken at discrete real-time intervals. The Extended Kalman Filter (EKF) approach modifies the standard Kalman filter (used for linear systems) in order to treat noisy nonlinear systems. It starts with an initial guess, and then updates this predicted state continually with new measurements. Unfortunately, if disturbances are so large that linearization is inadequate to describe the system, the filter will not converge to a reasonable estimate. First, we give a brief outline of the Kalman filter and of the extended Kalman filter; then, using this background we will explain the rotor and motor extended Kalman filters. For a more complete explanation, the reader is referred to [11].

15.5.1 The Kalman Filter

Let us describe a dynamic system using a linear-difference-state equation, as follows:

$$\boldsymbol{X}_i = \boldsymbol{\Phi}_{i/i-1}\boldsymbol{X}_{i-1} + \boldsymbol{W}_i. \tag{15.22}$$

The state of the system at t_i is given by the n-dimensional vector $\boldsymbol{X}_i$. The term $\boldsymbol{\Phi}_{i/i-1}$ is an $n \times n$ transition matrix and $\boldsymbol{W}_i$ is the random error with the known first- and second-order characteristics

$$E[\boldsymbol{W}_i] = \boldsymbol{0}, \qquad i = 0, 1, \ldots \tag{15.23}$$

$$E[\boldsymbol{W}_i \boldsymbol{W}_j^T] = \boldsymbol{\mathcal{Q}}_i \delta_{ij}, \, In, \tag{15.24}$$

where δ_{ij} is the Kronecker delta function. The matrix $\boldsymbol{\mathcal{Q}}_i$ is assumed to be Extended Kalman Filter (EKF).

Suppose that at each time t_i there is available an m-dimensional vector of measurement $\boldsymbol{Z}_i$ that is linearly related to the state and which is corrupted by the additive noise $\boldsymbol{V}_i$. This is the so-called observation equation

$$\boldsymbol{Z}_i = \boldsymbol{\mathcal{H}}_i \boldsymbol{X}_i + \boldsymbol{V}_i, \tag{15.25}$$

where $\boldsymbol{\mathcal{H}}_i$ is a known $m \times n$ observation matrix and the vector $\boldsymbol{V}_i$ is a random error with the known statistics

$$E[\boldsymbol{V}_i] = \boldsymbol{0}, \qquad i = 0, 1, \ldots \tag{15.26}$$

$$E[\boldsymbol{V}_i \boldsymbol{V}_j^T] = \boldsymbol{\mathcal{C}}_i \delta_{ij}, \tag{15.27}$$

where the matrix $\boldsymbol{\mathcal{C}}_i$ is assumed to be nonnegative definite [11].

Further, assume that the random processes $\boldsymbol{W}_i$ and $\boldsymbol{V}_i$ are uncorrelated, that is, for each $i, \; j$,

$$E[\boldsymbol{W}_i \boldsymbol{V}_j^T] = \boldsymbol{\mathcal{O}}, \tag{15.28}$$

where $\boldsymbol{\mathcal{O}}$ is the zero matrix.

Given the preceding models (15.22) and (15.25), we shall determine an estimate $\hat{\boldsymbol{X}}_i$ of the state at t_i that is a linear combination of an estimate $\hat{\boldsymbol{X}}_{i-1}$ at t_{i-1} and the data $\boldsymbol{Z}_i$ measured at time t_i. By defining an unknown $(n \times m)$ gain matrix $\boldsymbol{\mathcal{K}}_i$, the estimate $\hat{\boldsymbol{X}}_i$ is given by

$$\hat{\boldsymbol{X}}_i = \boldsymbol{\Phi}_{i/i-1} \hat{\boldsymbol{X}}_{i-1} + \boldsymbol{\mathcal{K}}_i [\boldsymbol{Z}_i - \boldsymbol{\mathcal{H}}_i \boldsymbol{\Phi}_{i/i-1} \hat{\boldsymbol{X}}_{i-1}]. \tag{15.29}$$

The matrix $\boldsymbol{\mathcal{K}}_i$ is determined so that the estimate has the minimal variance, that is, $\hat{\boldsymbol{X}}_i$ is chosen so as to minimize its mean squared error:

$$E_{MIN} = \{E[(\hat{\boldsymbol{X}}_i - \boldsymbol{X}_i)^T (\hat{\boldsymbol{X}}_i - \boldsymbol{X}_i)]\}_{MIN}. \tag{15.30}$$

Equation (15.30) is equivalent to the minimization of the trace of the state error covariance matrix $\boldsymbol{\mathcal{P}}_i$, that is,

$$E_{MIN} = \{trace \; \boldsymbol{\mathcal{P}}_i\}_{MIN} = \{trace \; E[(\hat{\boldsymbol{X}}_i - \boldsymbol{X}_i)(\hat{\boldsymbol{X}}_i - \boldsymbol{X}_i)^T]\}_{MIN}. \tag{15.31}$$

By substituting Eq. (15.25) into (15.29), and then substituting Eqs. (15.29) and (15.22) into Eq. (15.31), it can be shown that the trace of the matrix $\boldsymbol{\mathcal{P}}_i$ will be minimized by choosing the following optimal gain matrix $\boldsymbol{\mathcal{K}}_i$:

$$\mathcal{K}_i = \mathcal{P}_{i/i-1}\mathcal{H}_i^T(\mathcal{H}_i\mathcal{P}_{i/i-1}\mathcal{H}_i^T + \mathcal{C}_i)^{-1}, \tag{15.32}$$

where $\mathcal{P}_{i/i-1}$ is the error covariance matrix

$$\mathcal{P}_{i/i-1} = \Phi_{i/i-1}\mathcal{P}_i\Phi_{i/i-1}^T + \mathcal{Q}_i \tag{15.33}$$

of the predicted state

$$\hat{X}_{i/i-1} = \Phi_{i/i-1}\hat{X}_i. \tag{15.34}$$

With this optimal gain matrix $\mathcal{K}_i$, the matrix $\mathcal{P}_i$ reduces to

$$\mathcal{P}_i = \mathcal{P}_{i/i-1} - \mathcal{K}_i\mathcal{H}_i\mathcal{P}_{i/i-1} = (I - \mathcal{K}_i\mathcal{H}_i)\mathcal{P}_{i/i-1}. \tag{15.35}$$

Equations (15.29), (15.33), (15.32), and (15.35) constitute the Kalman filter equations for the model of the system (15.22) and that of the measurement (15.25), respectively.

From Eq. (15.32), we see that as the measurement error covariance matrix $\mathcal{C}_i$ approaches zero, the gain matrix $\mathcal{K}_i$ weights the residual second term of Eq. (15.29) more heavily:

$$\lim_{\mathcal{C}_i \to \mathcal{O}} \mathcal{K}_i = \mathcal{H}_i^{-1}. \tag{15.36}$$

On the other hand, as the estimated state error covariance $\mathcal{P}_i$ approaches zero, the gain $\mathcal{K}_i$ weights the residual second term of Eq. (15.29) less heavily:

$$\lim_{\mathcal{P}_i \to \mathcal{O}} \mathcal{K}_i = \mathcal{O}. \tag{15.37}$$

Another way of thinking about the control of the gain of the Kalman filter by $\mathcal{K}_i$ is to consider that as the measurement error covariance matrix $\mathcal{C}_i$ approaches zero, the actual measurement $\mathcal{Z}_i$ is "trusted" more and more, while the predicted state $\Phi_{i/i-1}\hat{X}_i$ is trusted less and less. On the other hand, as the estimated state error covariance $\mathcal{P}_i$ approaches zero the actual measurement $\mathcal{Z}_i$ is trusted less and less, while the predicted state $\Phi_{i/i-1}\hat{X}_i$ (the dynamic model) is trusted more and more.

15.5.2 The Extended Kalman Filter

As described previously, the Kalman filter addresses the general problem of trying to estimate the state X_i of a discrete-time controlled process that is governed by a linear stochastic difference equation. But what happens if the process and/or the relation between the measurement and the state is nonlinear? Some of the most interesting and successful applications of Kalman filtering are concerned with just these types of situations. A Kalman filter that linearizes about the current predicted state $\hat{X}_{i/i-1}$ and measurement $\mathcal{Z}_i$ is called an extended Kalman filter or EKF.

In computer vision, the measurement model is usually described by a nonlinear observation equation $\boldsymbol{f}_i(\boldsymbol{Z}_{0,i}, \boldsymbol{X}_i) = \boldsymbol{0}$, where the parameter $\boldsymbol{Z}_{0,i}$ is the accurate measurement. In practice, such a measurement is affected by random errors. We assume that the measurement system is disturbed by additive white noise

$$\boldsymbol{Z}_i = \boldsymbol{Z}_{0,i} + \boldsymbol{V}_i, \tag{15.38}$$

where the statistics of noise $\boldsymbol{V}_i$ is given by Eqs. (15.26) and (15.27).

To apply the Kalman filter technique, we must expand the nonlinear observation equation into a first-order Taylor series about $(\boldsymbol{Z}_i, \hat{\boldsymbol{X}}_{i/i-1})$,

$$\begin{aligned}\boldsymbol{f}_i(\boldsymbol{Z}_{0,i}, \boldsymbol{X}_i) = \boldsymbol{f}_i(\boldsymbol{Z}_i, \hat{\boldsymbol{X}}_{i/i-1}) + \frac{\partial \boldsymbol{f}_i(\boldsymbol{Z}_i, \hat{\boldsymbol{X}}_{i/i-1})}{\partial \boldsymbol{Z}_{0,i}}(\boldsymbol{Z}_{0,i} - \boldsymbol{Z}_i) \\ + \frac{\partial \boldsymbol{f}_i(\boldsymbol{Z}_i, \hat{\boldsymbol{X}}_{i/i-1})}{\partial \boldsymbol{X}_i}(\boldsymbol{X}_i - \hat{\boldsymbol{X}}_{i/i-1}) + \mathcal{O}^2 = \boldsymbol{0}.\end{aligned} \tag{15.39}$$

By ignoring the second-order term $\mathcal{O}^2$, the linearized measurement Eq. (15.39) becomes

$$\boldsymbol{Y}_i = \boldsymbol{\mathcal{H}}_i \boldsymbol{X}_i + \boldsymbol{N}_i, \tag{15.40}$$

where $\boldsymbol{Y}_i$ is the new measurement vector, $\boldsymbol{N}_i$ is the noise vector of the new measurement, and $\boldsymbol{\mathcal{H}}_i$ is the linearized transformation matrix. The components of Eq. (15.40) are given by

$$\begin{aligned}\boldsymbol{Y}_i &= -\boldsymbol{f}_i(\boldsymbol{Z}_i, \hat{\boldsymbol{X}}_{i/i-1}) + \frac{\partial \boldsymbol{f}_i(\boldsymbol{Z}_i, \hat{\boldsymbol{X}}_{i/i-1})}{\partial \boldsymbol{X}_i}\hat{\boldsymbol{X}}_{i/i-1}, \\ \boldsymbol{\mathcal{H}}_i &= \frac{\partial \boldsymbol{f}_i(\boldsymbol{Z}_i, \hat{\boldsymbol{X}}_{i/i-1})}{\partial \boldsymbol{X}_i}, \\ \boldsymbol{N}_i &= \frac{\partial \boldsymbol{f}_i(\boldsymbol{Z}_i, \hat{\boldsymbol{X}}_{i/i-1})}{\partial \boldsymbol{Z}_{0,i}}(\boldsymbol{Z}_{0,i} - \boldsymbol{Z}_i), \\ E[\boldsymbol{N}_i] &= \boldsymbol{0}, \\ E[\boldsymbol{N}_i \boldsymbol{N}_i^T] &= \boldsymbol{\mathcal{C}}_{i/i-1} = \frac{\partial \boldsymbol{f}_i(\boldsymbol{Z}_i, \hat{\boldsymbol{X}}_{i/i-1})}{\partial \boldsymbol{Z}_{0,i}}\boldsymbol{\mathcal{C}}_i\frac{\partial \boldsymbol{f}_i(\boldsymbol{Z}_i, \hat{\boldsymbol{X}}_{i/i-1})}{\partial \boldsymbol{Z}_{0,i}}^T,\end{aligned}$$

where $\boldsymbol{\mathcal{C}}_i$ is given by the statistics of the measurement (15.27). This linearized Eq. (15.40) is a general form for the nonlinear model. We will use this form for our particular nonlinear measurement model in Sect. 15.6.

15.5.3 The Rotor Extended Kalman Filter

This subsection describes an EKF algorithm to estimate rotors or quaternions. In the static case, the measurements of points free of error satisfy the conditions

$$\boldsymbol{R}_{i+1} = \boldsymbol{R}_i, \tag{15.41}$$

$$\boldsymbol{p}'_{0,i+1} = \boldsymbol{\mathcal{R}}(\boldsymbol{R}_{i+1})\boldsymbol{p}_{0,i+1}, \tag{15.42}$$

where $\{\boldsymbol{p}_{0,i}\}$ and $\{\boldsymbol{p}'_{0,i}\}$ are sets of points before and after rotation, respectively, and $\boldsymbol{R}_i$ is the rotation quaternion for the i-th pair of points $p_{0,i}$ and $p'_{0,i}$. $\boldsymbol{\mathcal{R}}(\boldsymbol{R})$ is the matrix representation of $\boldsymbol{R}$,

$$\boldsymbol{\mathcal{R}}(\boldsymbol{R}) = \begin{pmatrix} r_1^2 + r_2^2 - r_3^2 - r_4^2 & 2(r_2 r_3 + r_1 r_4) & 2(r_2 r_4 - r_1 r_3) \\ 2(r_2 r_3 - r_1 r_4) & r_1^2 - r_2^2 + r_3^2 - r_4^2 & 2(r_4 r_3 + r_1 r_2) \\ 2(r_2 r_4 + r_1 r_3) & 2(r_4 r_3 - r_1 r_2) & r_1^2 - r_2^2 - r_3^2 + r_4^2, \end{pmatrix}, \tag{15.43}$$

where r_j for $j = 1, 2, 3, 4$ represents the four components of $\boldsymbol{R}$ which satisfy the condition

$$\|\boldsymbol{R}\| = 1. \tag{15.44}$$

Let us assume that the measurements $\{\boldsymbol{p}_{i+1}\}$ and $\{\boldsymbol{p}'_{i+1}\}$ of $\{\boldsymbol{p}_{0,i+1}\}$ and $\{\boldsymbol{p}'_{0,i+1}\}$ are corrupted by noise $\{\boldsymbol{n}_{i+1}\}$ and $\{\boldsymbol{n}'_{i+1}\}$, respectively, such that

$$\boldsymbol{p}_{i+1} = \boldsymbol{p}_{0,i+1} + \boldsymbol{n}_{i+1} \tag{15.45}$$

$$\boldsymbol{p}'_{i+1} = \boldsymbol{p}'_{0,i+1} + \boldsymbol{n}'_{i+1}. \tag{15.46}$$

Here, it is assumed that the noise vectors $\{\boldsymbol{n}_{i+1}\}$ and $\{\boldsymbol{n}'_{i+1}\}$ have zero mean and the respective covariance matrices $\{\boldsymbol{\mathcal{C}}_{i+1}\}$ and $\{\boldsymbol{\mathcal{C}}'_{i+1}\}$ are known.

We rewrite Eq. (15.42) as the function $\boldsymbol{f}_{i+1}$ depending on the variables ($\boldsymbol{p}_{0,i+1}$, $\boldsymbol{p}'_{0,i+1}$, $\boldsymbol{R}_{i+1}$):

$$\boldsymbol{f}_{i+1}(\boldsymbol{p}_{0,i+1}, \boldsymbol{p}'_{0,i+1}, \boldsymbol{R}_{i+1}) = \boldsymbol{p}'_{0,i+1} - \boldsymbol{\mathcal{R}}(\boldsymbol{R}_{i+1})\boldsymbol{p}_{0,i+1} = \boldsymbol{0}. \tag{15.47}$$

Expanding this equation about ($\boldsymbol{p}_{i+1}$, $\boldsymbol{p}'_{i+1}$, $\hat{\boldsymbol{R}}_{i+1/i}$) in terms of the first-order Taylor series, we get

$$\boldsymbol{f}_{i+1}(\boldsymbol{p}_{0,i+1}, \boldsymbol{p}'_{0,i+1}, \boldsymbol{R}_{i+1}) = \boldsymbol{f}_{i+1}(\boldsymbol{p}_{i+1}, \boldsymbol{p}'_{i+1}, \hat{\boldsymbol{R}}_{i+1/i}) + \\ + \frac{\partial \boldsymbol{f}_{i+1}(\boldsymbol{p}_{i+1}, \boldsymbol{p}'_{i+1}, \hat{\boldsymbol{R}}_{i+1/i})}{\partial \boldsymbol{p}'_{0,i+1}}(\boldsymbol{p}'_{0,i+1} - \boldsymbol{p}'_{i+1}) +$$

$$+\frac{\partial \boldsymbol{f}_{i+1}(\boldsymbol{p}_{i+1}, \boldsymbol{p}'_{i+1}, \hat{\boldsymbol{R}}_{i+1/i})}{\partial \boldsymbol{p}_{0,i+1}}(\boldsymbol{p}_{0,i+1} - \boldsymbol{p}_{i+1}) + \tag{15.48}$$
$$+\frac{\partial \boldsymbol{f}_{i+1}(\boldsymbol{p}_{i+1}, \boldsymbol{p}'_{i+1}, \hat{\boldsymbol{R}}_{i+1/i})}{\partial \boldsymbol{R}_{i+1}}(\boldsymbol{R}_{i+1} - \hat{\boldsymbol{R}}_{i+1/i}) + \mathcal{O}^2 = \boldsymbol{0},$$

where the second-order term $\mathcal{O}^2$ can be omitted, and therefore

$$\frac{\partial \boldsymbol{f}_{i+1}(\boldsymbol{p}_{i+1}, \boldsymbol{p}'_{i+1}, \hat{\boldsymbol{R}}_{i+1/i})}{\partial \boldsymbol{p}'_{0,i+1}} = 1, \tag{15.49}$$
$$\frac{\partial \boldsymbol{f}_{i+1}(\boldsymbol{p}_{i+1}, \boldsymbol{p}'_{i+1}, \hat{\boldsymbol{R}}_{i+1/i})}{\partial \boldsymbol{p}_{0,i+1}} = -\boldsymbol{\mathcal{R}}(\hat{\boldsymbol{R}}_{i+1/i}), \tag{15.50}$$
$$\frac{\partial \boldsymbol{f}_{i+1}(\boldsymbol{p}_{i+1}, \boldsymbol{p}'_{i+1}, \hat{\boldsymbol{R}}_{i+1/i})}{\partial \boldsymbol{R}_{i+1}} = \frac{\partial \boldsymbol{\mathcal{R}}(\hat{\boldsymbol{R}}_{i+1/i})\boldsymbol{p}_{i+1}}{\partial \boldsymbol{R}_{i+1}}. \tag{15.51}$$

In order to compute expression (15.51), we utilize the following vectors:

$$\boldsymbol{p}_{i+1} \cong (p_1 \quad p_2 \quad p_3)^T, \tag{15.52}$$
$$\boldsymbol{R}_{i+1} \cong (r_1 \quad r_2 \quad r_3 \quad r_4)^T \tag{15.53}$$
$$\hat{\boldsymbol{R}}_{i+1/i} \cong \left(\hat{r}_1 \quad \hat{r}_2 \quad \hat{r}_3 \quad \hat{r}_4\right)^T. \tag{15.54}$$

Thus, we can write

$$\boldsymbol{\mathcal{R}}(\hat{\boldsymbol{R}}_{i+1/i})\boldsymbol{p}_{i+1} =$$
$$\begin{pmatrix} \hat{r}_1^2 + \hat{r}_2^2 - \hat{r}_3^2 - \hat{r}_4^2 & 2(\hat{r}_2\hat{r}_3 + \hat{r}_1\hat{r}_4) & 2(\hat{r}_2\hat{r}_4 - \hat{r}_1\hat{r}_3) \\ 2(\hat{r}_2\hat{r}_3 - \hat{r}_1\hat{r}_4) & \hat{r}_1^2 - \hat{r}_2^2 + \hat{r}_3^2 - \hat{r}_4^2 & 2(\hat{r}_4\hat{r}_3 + \hat{r}_1\hat{r}_2) \\ 2(\hat{r}_2\hat{r}_4 + \hat{r}_1\hat{r}_3) & 2(\hat{r}_4\hat{r}_3 - \hat{r}_1\hat{r}_2) & \hat{r}_1^2 - \hat{r}_2^2 - \hat{r}_3^2 + \hat{r}_4^2 \end{pmatrix} \begin{pmatrix} p_1 \\ p_2 \\ p_3 \end{pmatrix}$$
$$= \left(\hat{p}'_1 \quad \hat{p}'_2 \quad \hat{p}'_3\right)^T = \hat{\boldsymbol{p}}'_{i+1/i}.$$

The derivative of the vector $\hat{\boldsymbol{p}}'_{i+1/i}$ with respect to a vector $\boldsymbol{R}_{i+1}$ is of the form

$$\frac{\partial \boldsymbol{\mathcal{R}}(\hat{\boldsymbol{R}}_{i+1/i})\boldsymbol{p}_{i+1}}{\partial \boldsymbol{R}_{i+1}} = \frac{\partial \hat{\boldsymbol{p}}'_{i+1/i}}{\partial \boldsymbol{R}_{i+1}} = \begin{pmatrix} \frac{\partial \hat{p}'_1}{\partial r_1} & \frac{\partial \hat{p}'_1}{\partial r_2} & \frac{\partial \hat{p}'_1}{\partial r_3} & \frac{\partial \hat{p}'_1}{\partial r_4} \\ \frac{\partial \hat{p}'_2}{\partial r_1} & \frac{\partial \hat{p}'_2}{\partial r_2} & \frac{\partial \hat{p}'_2}{\partial r_3} & \frac{\partial \hat{p}'_2}{\partial r_4} \\ \frac{\partial \hat{p}'_3}{\partial r_1} & \frac{\partial \hat{p}'_3}{\partial r_2} & \frac{\partial \hat{p}'_3}{\partial r_3} & \frac{\partial \hat{p}'_3}{\partial r_4} \end{pmatrix}. \tag{15.55}$$

Now, defining a 3×4 matrix $\boldsymbol{\mathcal{H}}_{i+1/i} = -\frac{\partial \boldsymbol{\mathcal{R}}(\hat{\boldsymbol{R}}_{i+1/i})\boldsymbol{p}_{i+1}}{\partial \boldsymbol{R}_{i+1}}$, we can write the previous equation as

$$-\boldsymbol{\mathcal{H}}_{i+1/i} \stackrel{\text{def}}{=} \frac{\partial \boldsymbol{\mathcal{R}}(\hat{\boldsymbol{R}}_{i+1/i})\boldsymbol{p}_{i+1}}{\partial \boldsymbol{R}_{i+1}} = \begin{pmatrix} h_1 & h_2 & h_3 & h_4 \\ h_4 & -h_3 & h_2 & -h_1 \\ -h_3 & -h_4 & h_1 & h_2 \end{pmatrix}, \tag{15.56}$$

where

$$h_1 = 2(\hat{r}_1 p_1 + \hat{r}_4 p_2 - \hat{r}_3 p_3), \qquad h_2 = 2(\hat{r}_2 p_1 + \hat{r}_3 p_2 + \hat{r}_4 p_3),$$
$$h_3 = 2(-\hat{r}_3 p_1 + \hat{r}_2 p_2 - \hat{r}_1 p_3), \qquad h_4 = 2(-\hat{r}_4 p_1 + \hat{r}_1 p_2 + \hat{r}_2 p_3).$$

By first substituting (15.56) into (15.51), and then substituting the resultant equation along with Eqs. (15.49) and (15.50) into Eq. (15.48), we get an expression in which the second-order terms can be omitted as follows:

$$\begin{aligned} \mathbf{0} = \boldsymbol{p}'_{i+1} - \mathcal{R}(\hat{\boldsymbol{R}}_{i+1/i})\boldsymbol{p}_{i+1} + (\boldsymbol{p}'_{0,i+1} - \boldsymbol{p}'_{i+1}) - \\ -\mathcal{R}(\hat{\boldsymbol{R}}_{i+1/i})(\boldsymbol{p}_{0,i+1} - \boldsymbol{p}_{i+1}) + \mathcal{H}_{i+1/i}(\boldsymbol{R}_{i+1} - \hat{\boldsymbol{R}}_{i+1/i}). \end{aligned} \tag{15.57}$$

This equation can then be further rearranged as

$$\begin{aligned} &\boldsymbol{p}'_{i+1} - \mathcal{R}(\hat{\boldsymbol{R}}_{i+1/i})\boldsymbol{p}_{i+1} + \mathcal{H}_{i+1/i}\hat{\boldsymbol{R}}_{i+1/i} \\ &= \mathcal{H}_{i+1/i}\boldsymbol{R}_{i+1} + (\boldsymbol{p}'_{i+1} - \boldsymbol{p}'_{0,i+1}) - \mathcal{R}(\hat{\boldsymbol{R}}_{i+1/i})(\boldsymbol{p}_{i+1} - \boldsymbol{p}_{0,i+1}). \end{aligned} \tag{15.58}$$

The terms of this equation are now identifiable as the measurement $\boldsymbol{z}_{i+1}$ and the noise of the measurement $\boldsymbol{n}_{i+1/i}$ as follows:

$$\boldsymbol{z}_{i+1} = \boldsymbol{p}'_{i+1} - \mathcal{R}(\hat{\boldsymbol{R}}_{i+1/i})\boldsymbol{p}_{i+1} + \mathcal{H}_{i+1/i}\hat{\boldsymbol{R}}_{i+1/i}, \tag{15.59}$$

$$\begin{aligned} \boldsymbol{n}_{i+1/i} &= (\boldsymbol{p}'_{i+1} - \boldsymbol{p}'_{0,i+1}) - \mathcal{R}(\hat{\boldsymbol{R}}_{i+1/i})(\boldsymbol{p}_{i+1} - \boldsymbol{p}_{0,i+1}) \\ &= \boldsymbol{n}'_{i+1} - \mathcal{R}(\hat{\boldsymbol{R}}_{i+1/i})\boldsymbol{n}_{i+1}. \end{aligned} \tag{15.60}$$

Using these variables we can finally write the first-order linearized measurement equation more compactly:

$$\boldsymbol{z}_{i+1} = \mathcal{H}_{i+1/i}\boldsymbol{R}_{i+1} + \boldsymbol{n}_{i+1/i}. \tag{15.61}$$

Here, $\boldsymbol{n}_{i+1/i}$ represents zero mean noise, with the covariance given by

$$\mathcal{C}_{i+1/i} = \mathcal{C}'_{i+1} + \mathcal{R}(\hat{\boldsymbol{R}}_{i+1/i})\mathcal{C}_{i+1}\mathcal{R}^T(\hat{\boldsymbol{R}}_{i+1/i}), \tag{15.62}$$

where $\mathcal{C}'_{i+1}$ and $\mathcal{C}_{i+1}$ are the known covariance matrices of noise $\boldsymbol{n}'_{i+1}$ and $\boldsymbol{n}_{i+1}$, respectively.

15.5.3.1 Rotation Estimation

Next, we will describe the procedure for estimating a rotation expressed as a rotor $\boldsymbol{R}$. At the beginning of the iteration, or step $i = 0$, the initial state $\hat{\boldsymbol{R}}_0$ and an initial estimation error covariance matrix $\mathcal{P}_0$ are given. According to Eq. (15.41), given an

estimate for $\hat{\boldsymbol{R}}^*{}_i$ and the covariance matrix $\boldsymbol{\mathcal{P}}_i$ at step i, it is reasonable to predict the next step $i+1$ in this way:

$$\hat{\boldsymbol{R}}_{i+1/i} = \hat{\boldsymbol{R}}^*{}_i, \tag{15.63}$$

$$\boldsymbol{\mathcal{P}}_{i+1/i} = \boldsymbol{\mathcal{P}}_i. \tag{15.64}$$

Taking into account the measurements $\boldsymbol{p}_{i+1}$, $\boldsymbol{p}'_{i+1}$ and the predicted state $\hat{\boldsymbol{R}}_{i+1/i}$, the new measurement z_{i+1} from Eq. (15.59) can be straightforwardly computed. Then the Kalman gain matrix and the estimate $\hat{\boldsymbol{R}}_{i+1}$ at step $i+1$ are computed using EKF as follows:

$$\boldsymbol{\mathcal{K}}_{i+1} = \boldsymbol{\mathcal{P}}_{i+1/i}\boldsymbol{\mathcal{H}}^T_{i+1/i}(\boldsymbol{\mathcal{H}}_{i+1/i}\boldsymbol{\mathcal{P}}_{i+1/i}\boldsymbol{\mathcal{H}}_{i+1/i} + \boldsymbol{\mathcal{C}}_{i+1/i})^{-1}, \tag{15.65}$$

$$\begin{aligned}\hat{\boldsymbol{R}}_{i+1} &= \hat{\boldsymbol{R}}_{i+1/i} + \boldsymbol{\mathcal{K}}_{i+1}(z_{i+1} - \boldsymbol{\mathcal{H}}_{i+1/i}\hat{\boldsymbol{R}}_{i+1/i}),\\ &= \hat{\boldsymbol{R}}_{i+1/i} + \boldsymbol{\mathcal{K}}_{i+1}(\boldsymbol{p}'_{i+1} - \boldsymbol{\mathcal{R}}(\hat{\boldsymbol{R}}_{i+1/i})\boldsymbol{p}_{i+1}).\end{aligned} \tag{15.66}$$

Prudently, $\hat{\boldsymbol{R}}_{i+1}$ must be modified to satisfy the constraint $\|\boldsymbol{R}\| = 1$:

$$\hat{\boldsymbol{R}}^*{}_{i+1} = \frac{\hat{\boldsymbol{R}}_{i+1}}{\|\hat{\boldsymbol{R}}_{i+1}\|}. \tag{15.67}$$

The updating of the error covariance matrix follows:

$$\boldsymbol{\mathcal{P}}_{i+1} = (I - \boldsymbol{\mathcal{K}}_{i+1}\boldsymbol{\mathcal{H}}_{i+1})\boldsymbol{\mathcal{P}}_{i+1/i}(I - \boldsymbol{\mathcal{K}}_{i+1}\boldsymbol{\mathcal{H}}_{i+1})^T + \boldsymbol{\mathcal{K}}_{i+1}\boldsymbol{\mathcal{C}}_{i+1}\boldsymbol{\mathcal{K}}^T_{i+1}. \tag{15.68}$$

Note that $\boldsymbol{\mathcal{H}}_{i+1}$ and $\boldsymbol{\mathcal{C}}_{i+1}$ are recalculated using the current estimate $\hat{\boldsymbol{R}}^*{}_{i+1}$. As far as the implementation of the filter is concerned, the measurement error covariance matrix $\boldsymbol{\mathcal{C}}_i$ might be measured prior to the operation of the filter.

15.6 The Motor Extended Kalman Filter

The use of the EKF filter in the motor algebra framework gives us the simultaneous estimation of translation and rotation. According to the literature, there are only batch methods for the simultaneous estimation of these components [7, 12]. The Motor Extended Kalman Filter (MEKF) turns out to be a natural extension of the rotor extended Kalman filter, thanks to the multivector concept of geometric algebra. First, let us define the noisy motion equation using lines in the motor algebra framework $\mathcal{G}^+_{1,3,0}$. The geometric features we will consider for the measurements are 3D observed lines ($\boldsymbol{L}^1, \boldsymbol{L}^2, \ldots, \boldsymbol{L}^n$, $n \geq 2$) which belong to an object moving in the 3D space. The rigid motion parameters between any pairing of consecutive time instants $(t_0, t_1, t_2, \ldots, t_N)$ are described compactly by the motor $\boldsymbol{M}_i$. According to Eq. (5.17), the motion of any line of the object is modeled by

$$\boldsymbol{L}_i = \boldsymbol{M}_i \boldsymbol{L}_{i-1} \widetilde{\boldsymbol{M}}_i. \tag{15.69}$$

If the change of the parameters of the line in motion between the time instants t_{i-1} and t_i is described in terms of motor velocity $\boldsymbol{V}_{i/i-1}$,

$$\boldsymbol{L}_i = \boldsymbol{V}_{i/i-1} \boldsymbol{L}_{i-1} \widetilde{\boldsymbol{V}}_{i/i-1}, \tag{15.70}$$

we can then express the recursive motion equation of the line, in general, as follows:

$$\boldsymbol{L}_i = \boldsymbol{M}_i \boldsymbol{L}_{i-1} \widetilde{\boldsymbol{M}}_i = (\boldsymbol{V}_{i/i-1} \boldsymbol{M}_{i-1}) \boldsymbol{L}_{i-1} (\widetilde{\boldsymbol{M}}_{i-1} \widetilde{\boldsymbol{V}}_{i/i-1}). \tag{15.71}$$

Thus, we obtain the ideal dynamic motion model in terms of the motors:

$$\boldsymbol{M}_i = \boldsymbol{V}_{i/i-1} \boldsymbol{M}_{i-1}. \tag{15.72}$$

For example, suppose the motion is a screw motion with rotation of constant angular velocity ω about an axis of a known line ($\boldsymbol{L}_s = \bar{\boldsymbol{r}} + I\boldsymbol{t}_c \wedge \bar{\boldsymbol{r}}$) and with constant translation velocity $\boldsymbol{v}_s$ along the axis line. If the data sampling is done at equidistant time intervals, then the time instants can be represented by integers, so the motor equation reads

$$\boldsymbol{V}_{i/i-1} = \boldsymbol{V} = (1 + I\boldsymbol{v}_s/2)(\cos(\omega/2) + \sin(\omega/2)\boldsymbol{L}_s). \tag{15.73}$$

Since in real work the relation between $\boldsymbol{M}_{i-1}$ and $\boldsymbol{M}_i$ is known only approximately, the real dynamic model of the noisy 3D motion is given by

$$\boldsymbol{M}_i = \boldsymbol{\mathcal{V}}_{i/i-1,Ml} \boldsymbol{M}_{i-1} + \boldsymbol{W}_i, \tag{15.74}$$

where the statistics of $\boldsymbol{W}_i$ are given by Eqs. (15.23) and (15.24). Note that $\boldsymbol{\mathcal{V}}_{i/i-1,Ml}$ signifies the "left-multiplication matrix" of motor $\boldsymbol{\mathcal{V}}_{i/i-1}$.

15.6.1 Representation of the Line Motion Model in Linear Algebra

The line motion model presented in the previous section uses geometric algebra $\mathcal{G}^+_{3,0,1}$. Because the EKF computer algorithm is implemented using the techniques of linear algebra, we should also formulate the line motion model $\boldsymbol{L}' = \boldsymbol{M}\boldsymbol{L}\tilde{\boldsymbol{M}}$ within the frame of linear algebra.

Let us start by considering rotor relationships. The multiplication of two rotors $\boldsymbol{U}$ and $\boldsymbol{V}$ in geometric algebra $\mathcal{G}^+_{3,0,1}$ may be represented by

$$\begin{aligned} \boldsymbol{W} &= \boldsymbol{U}\boldsymbol{V} = (u_0 + \boldsymbol{u})(v_0 + \boldsymbol{v}), \\ &= u_0 v_0 + \boldsymbol{u} \cdot \boldsymbol{v} + u_0 \boldsymbol{v} + v_0 \boldsymbol{u} + \boldsymbol{u} \wedge \boldsymbol{v}. \end{aligned} \tag{15.75}$$

Multiplication of these two rotors in linear algebra is represented by

$$\boldsymbol{W} = \boldsymbol{\mathcal{U}}_{Rl} \boldsymbol{V} = \boldsymbol{\mathcal{V}}_{Rr} \boldsymbol{U}, \tag{15.76}$$

where $\boldsymbol{U} = (u_0 \quad u_1 \quad u_2 \quad u_3)^T$, $\boldsymbol{V} = (v_0 \quad v_1 \quad v_2 \quad v_3)^T$ and

$$\boldsymbol{\mathcal{U}}_{Rl} = \begin{pmatrix} u_0 & -u_1 & -u_2 & -u_3 \\ u_1 & u_0 & u_3 & -u_2 \\ u_2 & -u_3 & u_0 & u_1 \\ u_3 & u_2 & -u_1 & u_0 \end{pmatrix}, \tag{15.77}$$

$$\boldsymbol{\mathcal{V}}_{Rr} = \begin{pmatrix} v_0 & -v_1 & -v_2 & -v_3 \\ v_1 & v_0 & -v_3 & v_2 \\ v_2 & v_3 & v_0 & -v_1 \\ v_3 & -v_2 & v_1 & v_0 \end{pmatrix}. \tag{15.78}$$

We call $\boldsymbol{\mathcal{U}}_{Rl}$ the "left-multiplication matrix of rotor $\boldsymbol{U}$" and $\boldsymbol{\mathcal{V}}_{Rr}$ the "right-multiplication matrix of rotor $\boldsymbol{V}$."

Multiplying $\boldsymbol{S} = \boldsymbol{U} + I\boldsymbol{U}'$ and $\boldsymbol{T} = \boldsymbol{V} + I\boldsymbol{V}'$ in geometric algebra $\mathcal{G}^+_{3,0,1}$ gives

$$\begin{aligned} \boldsymbol{Q} &= \boldsymbol{S}\boldsymbol{T} = (\boldsymbol{U} + I\boldsymbol{U}')(\boldsymbol{V} + I\boldsymbol{V}'), \\ &= \boldsymbol{U}\boldsymbol{V} + I(\boldsymbol{U}\boldsymbol{V}' + \boldsymbol{U}'\boldsymbol{V}), \end{aligned} \tag{15.79}$$

where $\boldsymbol{U}$, $\boldsymbol{U}'$, $\boldsymbol{V}$, and $\boldsymbol{V}'$ are all expressed in the form of rotors. Multiplication of these two motors in terms of matrices gives

$$\boldsymbol{Q} = \boldsymbol{\mathcal{S}}_{Ml} \boldsymbol{T} = \boldsymbol{\mathcal{T}}_{Mr} \boldsymbol{S}, \tag{15.80}$$

where

$$\begin{aligned} \boldsymbol{S} &= (u_0 \quad u_1 \quad u_2 \quad u_3 \quad u'_0 \quad u'_1 \quad u'_2 \quad u'_3)^T, \\ \boldsymbol{T} &= (v_0 \quad v_1 \quad v_2 \quad v_3 \quad v'_0 \quad v'_1 \quad v'_2 \quad v'_3)^T, \\ \boldsymbol{\mathcal{S}}_{Ml} &= \begin{pmatrix} \boldsymbol{\mathcal{U}}_{Rl} & \boldsymbol{0}_{4\times4} \\ \boldsymbol{\mathcal{U}}'_{Rl} & \boldsymbol{\mathcal{U}}_{Rl} \end{pmatrix}, \quad \boldsymbol{\mathcal{T}}_{Mr} = \begin{pmatrix} \boldsymbol{\mathcal{V}}_{Rr} & \boldsymbol{0}_{4\times4} \\ \boldsymbol{\mathcal{V}}'_{Rr} & \boldsymbol{\mathcal{V}}_{Rr} \end{pmatrix}. \end{aligned} \tag{15.81}$$

In this case, we call $\boldsymbol{\mathcal{S}}_{Ml}$ the "left-multiplication matrix of motor $\boldsymbol{S}$" and $\boldsymbol{\mathcal{T}}_{Mr}$ the "right-multiplication matrix of motor $\boldsymbol{T}$."

To convert the line motion model of Eq. (5.17) to matrix algebra, we can treat the real and dual components $\boldsymbol{n}$, $\boldsymbol{m}$, $\boldsymbol{n}'$, and $\boldsymbol{m}'$ of the lines $\boldsymbol{L}$ and $\boldsymbol{L}'$ as rotors with zero scalar. By multiplying, from the right, both sides of Eq. (5.17) by $\boldsymbol{M}$, we get

$$\boldsymbol{L}'\boldsymbol{M} - \boldsymbol{M}\boldsymbol{L} = 0. \tag{15.82}$$

This results in the following linear motion equation:

$$(\boldsymbol{\mathcal{L}}'_{Ml} - \boldsymbol{\mathcal{L}}_{Mr})\boldsymbol{M} = \boldsymbol{\mathcal{A}}_M \boldsymbol{M} = \boldsymbol{0}. \tag{15.83}$$

This matrix representation of the line motion was suggested in [13]. The constraints of Eqs. (4.16) and (4.58), respectively, now are

$$\boldsymbol{R}^T \boldsymbol{R} = 1, \tag{15.84}$$
$$\boldsymbol{R}^T \boldsymbol{R}' = 0, \tag{15.85}$$

with $\boldsymbol{R} = (r_0\ r_1\ r_2\ r_3)^T$, $\boldsymbol{R}' = (r'_0\ r'_1\ r'_2\ r'_3)^T$, and $\boldsymbol{M} = \boldsymbol{R} + I\,\boldsymbol{R}'$.

These properties will be used for the implementation of the MEKF algorithm in the next section.

15.6.2 Linearization of the Measurement Model

Considering Eq. (15.83), we can easily see that the relation between the measurement $\boldsymbol{\mathcal{A}}_M$ and the state $\boldsymbol{M}$ is, unfortunately, nonlinear; that is why we should linearize it. Assume that the measurement $\boldsymbol{\mathcal{A}}_{\boldsymbol{M}i}$ represents the true data $\boldsymbol{\mathcal{A}}_{\boldsymbol{M}0,i}$ contaminated by measurement noise $\boldsymbol{\mathcal{N}}_{\mathcal{A}_M,i}$ with zero mean and known covariance matrix $\boldsymbol{\mathcal{C}}_{\mathcal{A}_M,i}$. Then,

$$\boldsymbol{\mathcal{A}}_{\boldsymbol{M}i} = \boldsymbol{\mathcal{A}}_{\boldsymbol{M}0,i} + \boldsymbol{\mathcal{N}}_{\mathcal{A}_M,i}. \tag{15.86}$$

Supposing that the predicted state of $\boldsymbol{M}_i$ is $\hat{\boldsymbol{M}}_{i/i}$, then according to Eq. (15.48), we can define a function $\boldsymbol{f}_{M,i}$ depending on the variables $(\boldsymbol{\mathcal{A}}_{\boldsymbol{M}0,i}, \boldsymbol{M}_i)$ as follows:

$$\boldsymbol{f}_{M,i}(\boldsymbol{\mathcal{A}}_{\boldsymbol{M}0,i}, \boldsymbol{M}_i) = \boldsymbol{\mathcal{A}}_{\boldsymbol{M}0,i}\boldsymbol{M}_i = \boldsymbol{0}. \tag{15.87}$$

This equation can then be expanded into a first-order Taylor series about the predicted state
$(\boldsymbol{\mathcal{A}}_{\boldsymbol{M}i}, \hat{\boldsymbol{M}}_{i/i-1})$:

$$\begin{aligned}\boldsymbol{f}_{M,i}(\boldsymbol{\mathcal{A}}_{\boldsymbol{M}0,i}, \boldsymbol{M}_i) = {} & \boldsymbol{f}_{M,i}(\boldsymbol{\mathcal{A}}_{M_i}, \hat{\boldsymbol{M}}_{i/i-1}) \\ & + \frac{\partial \boldsymbol{f}_{M,i}(\boldsymbol{\mathcal{A}}_{\boldsymbol{M}i}, \hat{\boldsymbol{M}}_{i/i-1})}{\partial \boldsymbol{M}_i}(\boldsymbol{M}_i - \hat{\boldsymbol{M}}_{i/i-1}) \\ & + (\boldsymbol{\mathcal{A}}_{\boldsymbol{M}0,i} - \boldsymbol{\mathcal{A}}_{\boldsymbol{M}i})\frac{\partial \boldsymbol{f}_{M,i}(\boldsymbol{\mathcal{A}}_{\boldsymbol{M}i}, \hat{\boldsymbol{M}}_{i/i-1})}{\partial \boldsymbol{\mathcal{A}}_{\boldsymbol{M}0,i}} + \mathcal{O}^2 = \boldsymbol{0}.\end{aligned} \tag{15.88}$$

Now, by renaming the components

$$\frac{\partial \boldsymbol{f}_{M,i}(\boldsymbol{\mathcal{A}}_{\boldsymbol{M}i}, \hat{\boldsymbol{M}}_{i/i-1})}{\partial \boldsymbol{M}_i} = \boldsymbol{\mathcal{A}}_{\boldsymbol{M}i}, \tag{15.89}$$

$$\frac{\partial \boldsymbol{f}_{M,i}(\boldsymbol{\mathcal{A}}_{\boldsymbol{M}i}, \hat{\boldsymbol{M}}_{i/i-1})}{\partial \boldsymbol{\mathcal{A}}_{\boldsymbol{M}0,i}} = \hat{\boldsymbol{M}}_{i/i-1}, \tag{15.90}$$

then omitting the second-order terms $\mathcal{O}^2$, and, finally, by taking into account Eq. (15.86) for $\boldsymbol{\mathcal{A}}_{\boldsymbol{M}i+1}$, we obtain

$$\begin{aligned}\boldsymbol{\mathcal{A}}_{\boldsymbol{M}i}\hat{\boldsymbol{M}}_{i/i-1} + \boldsymbol{\mathcal{A}}_{\boldsymbol{M}i}(\boldsymbol{M}_i - \hat{\boldsymbol{M}}_{i/i-1}) + (\boldsymbol{\mathcal{A}}_{\boldsymbol{M}0,i} - \boldsymbol{\mathcal{A}}_{\boldsymbol{M}i})\hat{\boldsymbol{M}}_{i/i-1} = \\ = \boldsymbol{\mathcal{A}}_{\boldsymbol{M}i}\hat{\boldsymbol{M}}_{i/i-1} + \boldsymbol{\mathcal{A}}_{\boldsymbol{M}i}(\boldsymbol{M}_i - \hat{\boldsymbol{M}}_{i/i-1}) - \boldsymbol{\mathcal{N}}_{\mathcal{A}_{M,i+1}}\hat{\boldsymbol{M}}_{i/i-1} \\ = \boldsymbol{0},\end{aligned} \tag{15.91}$$

or

$$-\boldsymbol{\mathcal{A}}_{\boldsymbol{M}i}\boldsymbol{M}_i + \boldsymbol{\mathcal{N}}_{\mathcal{A}_{M,i}}\hat{\boldsymbol{M}}_{i/i-1} = \boldsymbol{\mathcal{A}}_{\boldsymbol{M}i}\hat{\boldsymbol{M}}_{i/i-1} - \boldsymbol{\mathcal{A}}_{\boldsymbol{M}i}\hat{\boldsymbol{M}}_{i/i-1} = 0. \tag{15.92}$$

As a result, we can claim that the measurement equation for the MEKF at step i is given by

$$\boldsymbol{Z}_i = -\boldsymbol{\mathcal{A}}_{\boldsymbol{M}i}\boldsymbol{M}_i + \boldsymbol{\mathcal{N}}_{\mathcal{A}_M,i}\hat{\boldsymbol{M}}_{i/i-1} = \boldsymbol{\mathcal{H}}_i\boldsymbol{M}_i + \boldsymbol{N}_{Z,i} = \boldsymbol{0}, \tag{15.93}$$

where we call $\boldsymbol{\mathcal{H}}_i = -\boldsymbol{\mathcal{A}}_{M_i}$ and $\boldsymbol{N}_{Z,i} = \boldsymbol{\mathcal{N}}_{\mathcal{A}_{M,i+1}}\hat{\boldsymbol{M}}_{i/i-1}$. The covariance matrix of $\boldsymbol{N}_{Z,i}$ is $\boldsymbol{\mathcal{C}}_i$.

15.6.3 Enforcing a Geometric Constraint

In order to estimate the motor state, we assume first that at the beginning, or step $i = 0$, the initial state $\hat{\boldsymbol{M}}_{1/0}$ and the initial error covariance matrix of the estimate $\boldsymbol{\mathcal{P}}_{1/0}$ are given. Now, according to Eqs. (15.29), (15.33), (15.32), and (15.35), the estimation equation of the motor state is given by

$$\begin{aligned}\boldsymbol{M}_i^* &= \boldsymbol{\Phi}_{i/i-1}\hat{\boldsymbol{M}}_{i-1} + \boldsymbol{\mathcal{K}}_i(\boldsymbol{Z}_i - \boldsymbol{\mathcal{H}}_i\boldsymbol{\Phi}_{i/i-1}\hat{\boldsymbol{M}}_{i-1}), \\ &= \boldsymbol{\mathcal{V}}_{i/i-1,Ml}\hat{\boldsymbol{M}}_{i-1} + \boldsymbol{\mathcal{K}}_i(\boldsymbol{Z}_i - \boldsymbol{\mathcal{H}}_i\boldsymbol{\mathcal{V}}_{i/i-1,Ml}\hat{\boldsymbol{M}}_{i-1}), \\ &= (\boldsymbol{R}_i^{*T} \quad \boldsymbol{R'}_i^{*T})^T,\end{aligned} \tag{15.94}$$

where the optimal Kalman gain matrix $\boldsymbol{\mathcal{K}}_i$ is computed according to the formula

$$\boldsymbol{\mathcal{K}}_i = \boldsymbol{\mathcal{P}}_{i/i-1}\boldsymbol{\mathcal{H}}_i^T(\boldsymbol{\mathcal{H}}_i\boldsymbol{\mathcal{P}}_{i/i-1}\boldsymbol{\mathcal{H}}_i^T + \boldsymbol{\mathcal{C}}_i)^{-1}, \tag{15.95}$$

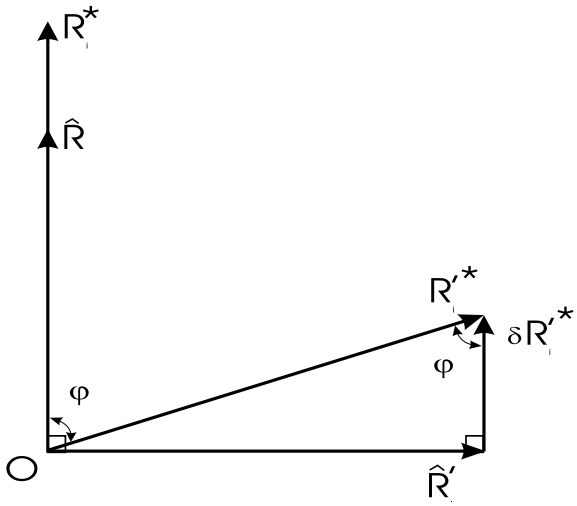


Fig. 15.6 Constraint of orthogonality according to $\hat{\boldsymbol{R}}\hat{\boldsymbol{R}}' = 0$

where

$$\boldsymbol{\mathcal{P}}_{i/i-1} = \boldsymbol{\Phi}_{i/i-1}\boldsymbol{\mathcal{P}}_i\boldsymbol{\Phi}_{i/i-1}^T + \boldsymbol{\mathcal{Q}}_i \tag{15.96}$$

and the error covariance matrix for the i-step is updated as

$$\boldsymbol{\mathcal{P}}_i = \boldsymbol{\mathcal{P}}_{i/i-1} - \boldsymbol{\mathcal{K}}_i\boldsymbol{\mathcal{H}}_i\boldsymbol{\mathcal{P}}_{i/i-1} = (\boldsymbol{I} - \boldsymbol{\mathcal{K}}_i\boldsymbol{\mathcal{H}}_i)\boldsymbol{\mathcal{P}}_{i/i-1}. \tag{15.97}$$

Now, $\boldsymbol{M}_i^*$ consists of two four-dimensional vectors $\boldsymbol{R}_i^*$ and $\boldsymbol{R}_i'^*$ which must be varied to fulfill the constraints of Eqs. (15.84) and (15.85). In the case of Eq. (15.84), the modifications can be done simply by considering the unit rotor

$$\hat{\boldsymbol{R}}_i = \frac{\boldsymbol{R}_i^*}{\|\boldsymbol{R}_i^*\|}. \tag{15.98}$$

It is not that simple to satisfy the constraint of Eq. (15.85), however. The constraint

$$\boldsymbol{R}'^T\boldsymbol{R} = 0 \tag{15.99}$$

tells us that $\boldsymbol{R}$ must be orthogonal to the dual rotor $\boldsymbol{R}'$ and valid up to a scalar (see the role of this scalar in Eq. (15.104)). Unfortunately, in practice, the rotor estimate $\boldsymbol{R}_i^*$ is usually not orthogonal to the estimated dual rotor $\boldsymbol{R}_i'^*$. Figure 15.6 suggests clearly how to enforce this geometric constraint in order to modify the orientation of $\boldsymbol{R}_i'^*$.

In order to do so, first we consider the cosine of the angle φ between estimates $\boldsymbol{R}_i^*$ and $\boldsymbol{R}_i'^*$:

$$\cos(\varphi) = \frac{\boldsymbol{R'^{*T}_i}\boldsymbol{R^*_i}}{\|\boldsymbol{R'^*_i}\| \cdot \|\boldsymbol{R^*_i}\|}. \tag{15.100}$$

This equation can be simplified by using Eq. (15.98) and the unit rotor $\hat{\boldsymbol{R}}_i$, as follows:

$$\cos(\varphi) = \frac{\boldsymbol{R'^{*T}_i}\hat{\boldsymbol{R}}_i}{\|\boldsymbol{R'^*_i}\|}. \tag{15.101}$$

Then we consider the deviation from the ideal orthogonal $\hat{\mathbf{R}}'$,

$$\delta\hat{\boldsymbol{R}}'^*_i = \|\hat{\boldsymbol{R}}'^*_i\|\cos(\varphi)\hat{\boldsymbol{R}}_i = (\boldsymbol{R'^{*T}_i}\hat{\boldsymbol{R}}_i)\boldsymbol{R}_i. \tag{15.102}$$

Finally, it is a straightforward matter to compute the ideal orthogonal $\hat{\boldsymbol{R}}'_i$, first we compute the unit rotor $\boldsymbol{R}_u$ orthogonal to $\hat{\boldsymbol{R}}_i$:

$$\boldsymbol{R}_u = \frac{(\boldsymbol{R'^*_i} - \delta\hat{\boldsymbol{R}}'^*_i)}{\|\boldsymbol{R'^*_i} - \delta\hat{\boldsymbol{R}}'^*_i\|} = \frac{(\boldsymbol{R'^*_i} - (\boldsymbol{R'^{*T}_i}\hat{\boldsymbol{R}}_i)\hat{\boldsymbol{R}}_i)}{\|\boldsymbol{R'^*_i} - (\boldsymbol{R'^{*T}_i}\hat{\boldsymbol{R}}_i)\hat{\boldsymbol{R}}_i\|}, \tag{15.103}$$

then we multiply it times the absolute value of the estimated $\boldsymbol{R'^*_i}$:

$$\hat{\boldsymbol{R}}'_i = \|\boldsymbol{R'^*_i}\|\boldsymbol{R}_u. \tag{15.104}$$

In other words, we have rotated the estimated $\boldsymbol{R'^*_i}$ until it is orthogonal to $\hat{\boldsymbol{R}}_i$.

15.6.4 Operation of the MEKF Algorithm

The processing of information using the MEKF filter can be explained very simply by considering the block diagrams presented in Fig. 15.7a, b.

In general, during the MEKF cycle, represented by Fig. 15.7a, the updating phase uses a prediction and a corrected input measurement to actualize a new estimate, which, in turn, will be modified using a geometric constraint. This cycle continues for each new measurement into infinity, but the MEKF should stabilize to the estimated proper states after only a few iterations.

The Kalman gain $\boldsymbol{\mathcal{K}}_i$ can be calculated before the actual estimation is carried out since it does not depend at all on the measurement of $\boldsymbol{Z}_i$. The computation cycle for $\boldsymbol{\mathcal{K}}_i$, illustrated in Fig. 15.7b, would proceed as follows: Now the MEKF algorithm illustrated in Fig. 15.8 will be explained. For the initialization of the MEKF, we can use for the initial time instant i the known values of $\hat{\boldsymbol{M}}_{i-1}$ and $\boldsymbol{\mathcal{P}}_{i-1}$, or, if we do not know them, we can simply choose the trivial values $\hat{\boldsymbol{M}}_{i-1} = [1\ 0\ 0\ 0\ 0\ 0\ 0\ 0]^T$ and $\boldsymbol{\mathcal{P}}_{i-1} = \boldsymbol{I}_{8\times 8}$. After the initialization, the MEKF seeks to determine $\hat{\boldsymbol{M}}_i$ for some future instant of time. The computation procedure of the MEKF would proceed as follows:

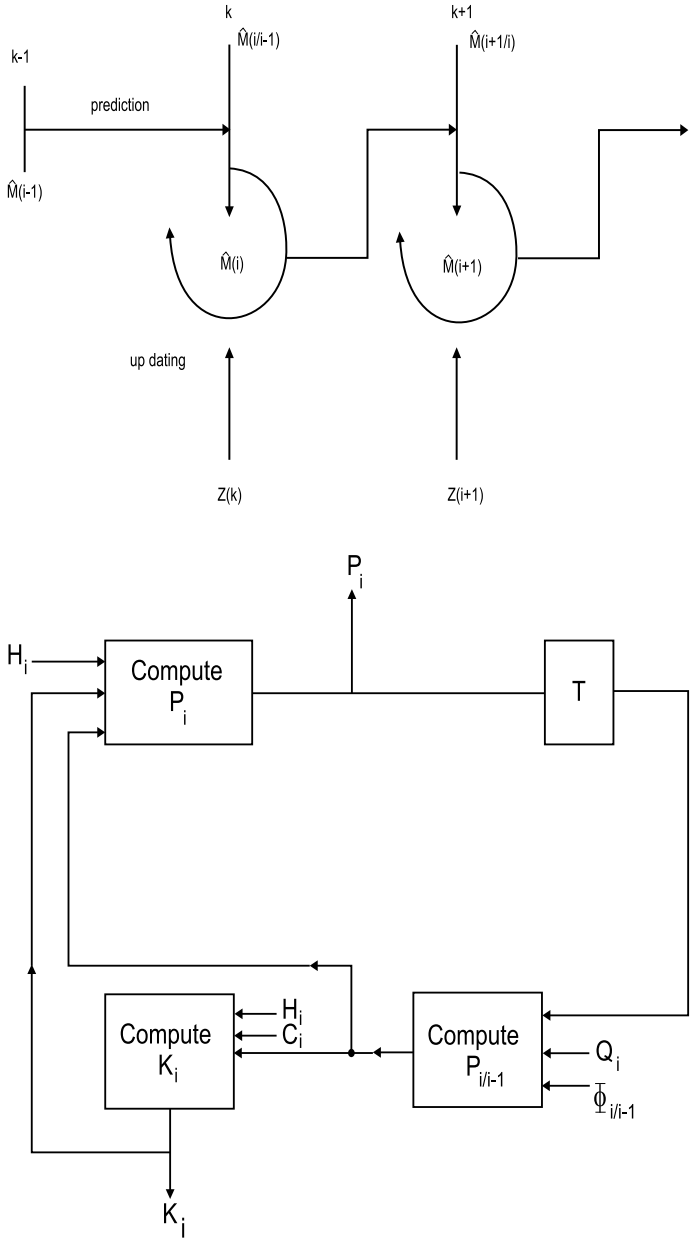


Fig. 15.7 MEKF operation: **a** cycles of estimation and updating and **b** Kalman gain computation

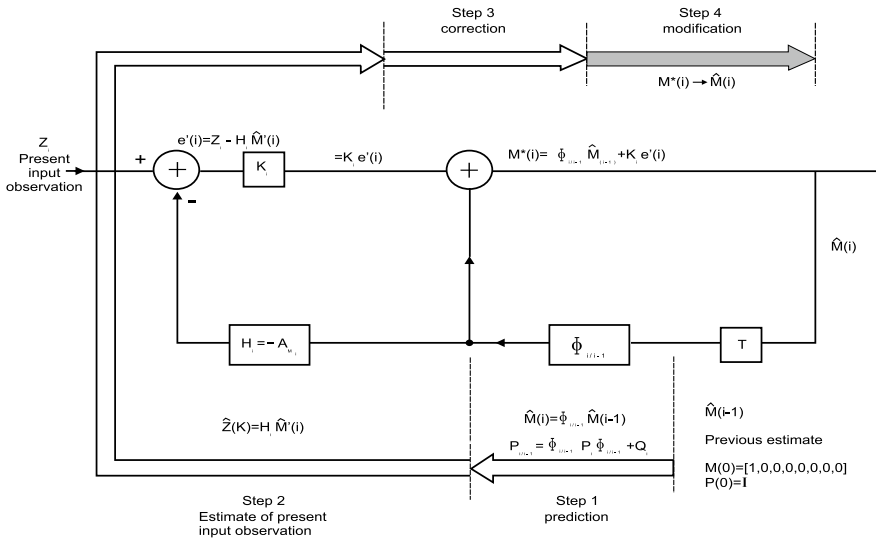


Fig. 15.8 Representation of the MEKF algorithm

Step	Procedure
1	Given $\mathcal{P}_{i-1/i-1} = \mathcal{P}_i$, $\mathcal{Q}_i$, and $\boldsymbol{\Phi}_{i/i-1}$, then $\mathcal{P}_{i/i-1}$ is computed using Eq. (15.96)
2	$\mathcal{P}_{i/i-1}$, $\mathcal{H}_i$, and $\mathcal{C}_i$ are substituted in Eq. (15.95) to obtain $\mathcal{K}_i$, which will be used in step 3 of the MEKF algorithm
3	$\mathcal{P}_{i/i-1}$, $\mathcal{K}_i$, and $\mathcal{H}_i$ are substituted in Eq. (15.97) to determine $\mathcal{P}_i$, which is stored until the time of the next measurement, when the cycle is repeated

Step	Action	Procedure
1	Prediction	The estimated $\hat{\boldsymbol{M}}_{i-1}$ is propagated forward by premultiplying it by the discrete system model matrix $\boldsymbol{\Phi}_{i/i-1}$. This gives the predicted estimate $\hat{\boldsymbol{M}}_{i/i-1} = \boldsymbol{\Phi}_{i/i-1}\hat{\boldsymbol{M}}_{i-1}$. Then the measurement model $\mathcal{H}_i = -\mathcal{A}_{\boldsymbol{M}_i}$ is linearized
2	Estimation	$\boldsymbol{\Phi}_{i/i-1}\hat{\boldsymbol{M}}_{i/i-1}$ is premultiplied by $\mathcal{H}_i = -\mathcal{A}_{\boldsymbol{M}_i}$, giving the estimated input measurement $\hat{\boldsymbol{Z}}_i = \mathcal{H}_i\boldsymbol{\Phi}_{i/i-1}\hat{\boldsymbol{M}}_{i/i-1}$, which is then subtracted from the actual measurement $\boldsymbol{Z}_i$ to obtain the measured residual or error, $\boldsymbol{e}'_i = \boldsymbol{Z}_i - \mathcal{H}_i\boldsymbol{\Phi}_{i/i-1}\hat{\boldsymbol{M}}_{i-1}$
3	Correction	The error $\boldsymbol{e}'_i$ is premultiplied by the matrix $\mathcal{K}_i$ and the result is added to $\hat{\boldsymbol{M}}_{i/i-1}$ to give the current estimate $\boldsymbol{M}^*_i$ (see Eq. (15.94))
4	Modification	The component $\boldsymbol{R}'^*_i$ of the estimate $\boldsymbol{M}^*_i = (\boldsymbol{R}^{*T}_i, \boldsymbol{R}'^{*T}_i)^T$ is modified using a geometric constraint according to Eq. (15.104). Then the final estimate, $\hat{\boldsymbol{M}}_i = (\hat{\boldsymbol{R}}^T_i, \hat{\boldsymbol{R}}'^T_i)^T$, is stored until the next measurement is made, at which time the cycle is repeated

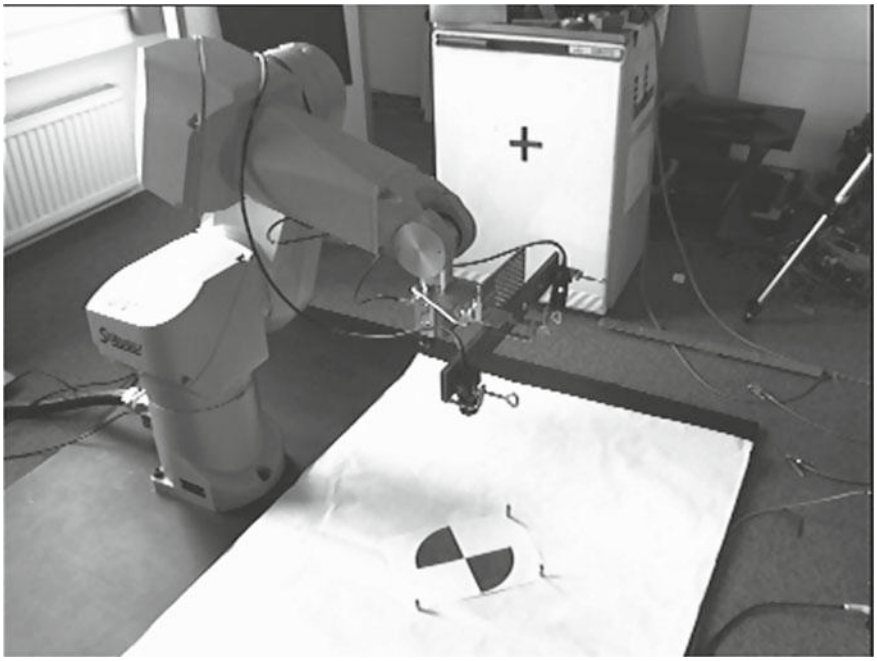


Fig. 15.9 Physical setup of the experiment

The MEKF would run recursively, completing these cycles until time instant N. It is necessary to mention that Kalman filter implementation is sensitive to numerical instability, but several techniques, such as square-root filtering and the so-called U-D factorization [14], are available to overcome such problems.

15.6.5 Estimation of the Relative Positioning of a Robot End-Effector

In this experiment, we applied the MEKF algorithm to estimate the relative motion between the end-joint of a Staubli RX90 robot arm and a 3D line belonging to a rigid object. The 3D line parameters were recovered during the arm movement using a stereo vision system. This approach can be used for several kinds of industrial applications, including maneuvering and grasping.

The physical setup of the experiment is shown in Fig. 15.9. The system looks to a pair of lines lying on the floor and moves in the 3D space, always conserving those lines in its field of view. The main task is to estimate automatically the relative motion between the floor lines and the system's end-joint. The visual system consists of two grayscale CCD 640×480 cameras fastened to the last joint of the robot arm.

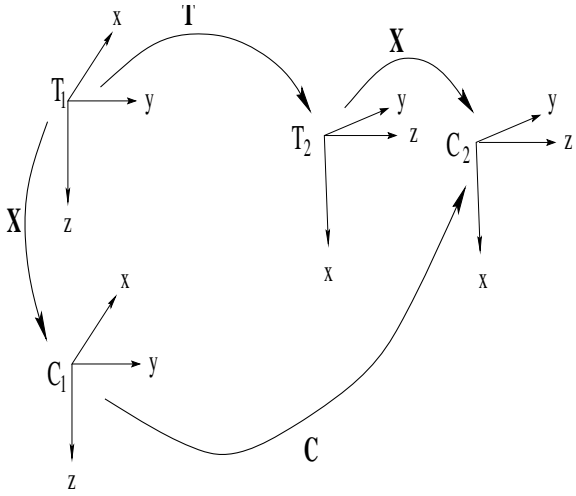


Fig. 15.10 Relationship between the tool system T and the camera system C

The Staubli robot arm has six joints which can be controlled by six variables (x, y, z, roll, pitch, and yaw). The coordinates (x, y, z) describe the position of the tool coordinate system T of the end-joint, which refers to the global coordinate system W of the robot. The orientation of the end-joint is described by the variables (roll, pitch, yaw) in terms of Euler angles. Since in practice we require three views in order to reconstruct a 3D line, we can create a virtual third line by slightly moving the two-camera stereo system, as illustrated in Fig. 15.10. The movement of the robot arm is controlled by the relative position and orientation between the tool coordinate system T and the base system W. The camera calibration procedure obtains the projective transformation matrix $\mathbf{P}$, which relates the visual space to the image plane up to a scalar value. The coordinate system of the camera C at the end-joint is related to the tool system frame T via a certain transformation $\mathbf{X}$ which was computed using a hand–eye calibration procedure [15]. When the tool system T is transformed from T_1 to T_2 by the transformation $\mathbf{T}$, the camera system C will be transformed from C_1 to C_2 by a certain transformation $\mathbf{C}$

$$\mathbf{C} = \mathbf{X}\mathbf{T}\mathbf{X}^{-1}. \tag{15.105}$$

Since the motion of the robot arm specified by the transformation $\mathbf{T}$ is known, we can compare it with the relative motion $\mathbf{C}$. The relative motion of the frame W and the frame C is a screw motion with constant angular velocity $w = -\phi/90$ and constant translation velocity $\boldsymbol{v}_s = 0.2$. The axis line $\boldsymbol{L}_s$ is parallel to the z-axis of the system C and one point touching this axis line is given by the coordinates (1.5, 0, 0). In motor algebra, $\mathcal{G}^+_{3,0,1}$, the axis line, is given by

$$\boldsymbol{L}_s = \gamma_1\gamma_2 + I(1.5\gamma_2\gamma_3) \wedge (\gamma_1\gamma_2) = \gamma_1\gamma_2 + I1.5\gamma_3\gamma_1, \tag{15.106}$$

Table 15.1 Reconstructed 3D lines

Time	Line	A point on the line	Direction
0	1	(0.000 3.087 −2.327)	(−0.345 0.937 −0.027)
	2	(0.556 0.000 −2.250)	(0.941 0.336 0.023)
1	1	(1.125 0.000 −2.027)	(−0.404 0.914 0.013)
	2	(0.701 0.000 −2.049)	(0.915 0.401 0.029)
2	1	(1.111 0.000 −1.82)	(−0.462 0.886 0.017)
	2	(0.794 0.000 −1.83)	(0.880 0.471 0.055)
⋮			
14	1	(0.018 0.000 0.648)	(−0.971 0.236 −0.036)
	2	(1.103 0.000 0.538)	(0.241 0.965 0.103)
15	1	(−0.680 0.000 0.753)	(−0.986 0.159 −0.025)
	2	(0.000 −6.341 0.783)	(0.171 0.985 −0.003)

and the motor $\boldsymbol{\mathcal{V}}$ is calculated as follows:

$$\begin{aligned} \boldsymbol{V} &= (1 + I\boldsymbol{v}_{s0}/2)(\cos(\omega/2) + \sin(\omega/2)\boldsymbol{L}_s) \\ &= 0.9994 - 0.0349\gamma_1\gamma_2 + I(0.0035 - 0.0523\gamma_3\gamma_1 + 0.0999\gamma_1\gamma_2). \end{aligned} \tag{15.107}$$

According to Eq. (15.72), the motor $\boldsymbol{M}_{i+1}$, expressed in terms of linear algebra, is given by

$$\boldsymbol{M}_i = \boldsymbol{\mathcal{V}}_{Ml}\boldsymbol{M}_{i-1}, \tag{15.108}$$

initialized with $\boldsymbol{M}_0 = (1\ 0\ 0\ 0\ 0\ 0\ 0\ 0)^T$. The reconstructed 3D lines listed in Table 15.1 were used to estimate the relative motion between the end-joint and the object on the floor using the MEKF algorithm.

The procedure followed in the experiment is summarized below: The algorithm

Step	Procedure
1	Cameras calibration to obtain $\boldsymbol{P}$ for each camera
2	Hand–eye calibration to obtain $\boldsymbol{X}$ for each camera
3	Robot arm movement images taken at constant sample rate (see Figs. 15.11 and 15.12)
4	Extraction of 2D lines from the images using Hough transform (see Figs. 15.11 and 15.12)
5	3D line reconstruction using matched lines of three images (see Table 15.1)
6	Estimation of the motion using 3D line observations and the MEKF algorithm (see Fig. 15.13)

for motion estimation runs online recursively following steps 3 to 6.

Fig. 15.11 Stereo triplet of sample object at time $I = 0$ with its edge images overlapped by extracted 2D lines

Fig. 15.12 Stereo triplet of a sample object at time $i = 4$ with its edge images overlapped by extracted 2D lines

Figure 15.13 presents the eight estimated parameters of the motion for 15 instants of time. We can see clearly that after only four observations the MEKF algorithm starts to follow almost perfectly the ground truth of the eight parameters. This real experiment together with the previous simulation confirms that the MEKF algorithm is an appropriate tool for the estimation of screw transformations using line observations.

15.7 Conclusion

In this chapter, we modeled the motion of lines in 4D space using motor algebra. This kind of modeling linearizes 3D Euclidean rigid motion transformation. The model of motion of lines using motors is very appealing for the design of an extended Kalman filter for the motion estimation. For the design of the filter, we started with the rotor extended Kalman filter. As a natural extension of that process, we then posited the theoretical foundations for a Motor Extended Kalman Filter (MEKF).

The MEKF algorithm has the virtue that it can estimate rotation and translation transformations simultaneously. Since most recursive algorithms in the literature compute translation and rotation transformations separately, we can claim that this is one of the most important advantages of the MEKF. Additionally, using the modeling of the lines in motor algebra we were able to linearize the nonlinear measurement model, thereby avoiding the problem of singularities. The dynamic motion model using motors as states is useful to effectively formulate and to compute the screw motion of a line as a minimal rigid entity. In the algorithm of MEKF, we modified the estimating step so that a certain geometric constraint is satisfied, which made the estimation converge faster to a proper motor state. Tests with simulated data [16] confirmed that the tuning of the MEKF parameters improved substantially the MEKF capabilities.

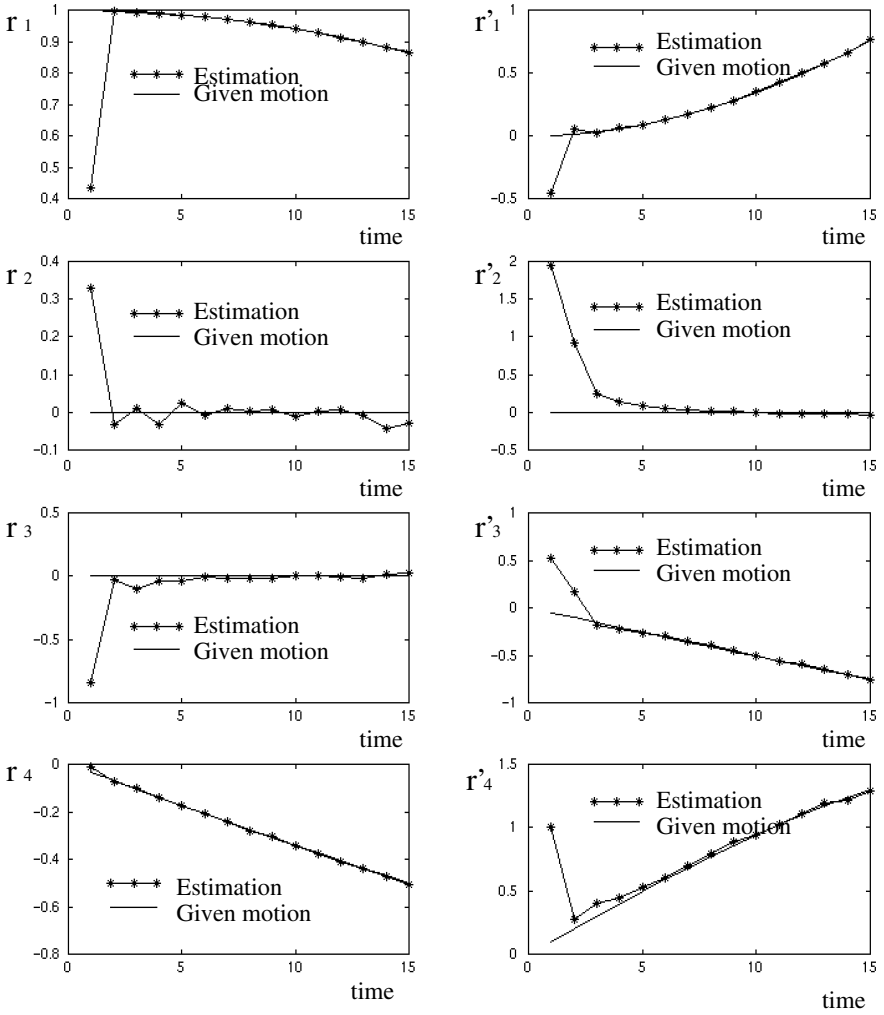


Fig. 15.13 Estimated motor parameters by the MEKF for the visually guided robot system

We presented a real application of visually guided robot manipulation. The system was efficiently calibrated using controlled robot movements and an effective hand–eye calibration method. The recovery of the parameters of the 3D lines was carried out using a stereo vision system and the techniques of Hough transform and shape filtering for the line matching. During robot maneuvering, the MEKF algorithm efficiently estimated the relative motion between its end-joint and a 3D object. These experiments confirmed that the new MEKF algorithm is an attractive online method for estimating screw motions using line observations.

References

1. Shiu, Y. C., & Ahmad, S. (1989). Calibration of wrist-mounted robotic sensors by solving homogeneous transform equations of the form $AX = XB$. *IEEE Transactions on Robotics Automation*, *5*, 16–27.
2. Tsai, R. Y., & Lenz, R. K. (1989). A new technique for fully autonomous and efficient 3D robotics hand/eye calibration. *IEEE Transactions on Robotics and Automation*, *5*, 345–358.
3. Wang, C. (1992). Extrinsic calibration of a vision sensor mounted on a robot. *IEEE Transactions on Robotics and Automation*, 161–175.
4. Chou, J. C. K., & Kamel, M. (1991). Finding the position and orientation of a sensor on a robot manipulator using quaternions. *International Journal of Robotics Research*, *10*(3), 240–254.
5. Gu, Y. L., & Luh, J. Y. S. (1987). Dual-number transformation and its applications to robotics. *IEEE Journal of Robotics and Automation*, *RA–3*(6), 615–623.
6. Li, M., & Betsis, D. (1995). Hand–eye calibration. In *Proceedings of the International Conference on Computer Vision* (pp. 40–46), Boston, June 20–23.
7. Horaud, R., & Dornaika, F. (1995). Hand-eye calibration. *International Journal of Robotics Research*, *14*, 195–210.
8. Chen, H. (1991). A screw motion approach to uniqueness analysis of head–eye geometry. In *IEEE Conference on Computer Vision and Pattern Recognition* (pp. 145–151), Maui, Hawaii, June 3–6.
9. Sabata, R., & Aggarwal, J. K. (1991). Estimation of motion from a pair of range images: A review. *CVGIP: Image Understanding*, *54*, 309–324.
10. Press, W. H., Teukolsky, S. A., Vetterling, W. T., & Flannery, B. P. (1994). *Numerical recipes in C*. New York: Cambridge University Press.
11. Sorenson, H. W. (1966). Kalman filtering techniques. In C. T. Leondes (Ed.), *Advances in control systems theory and applications* (Vol. 3, pp. 218–292). New York: Academic Press.
12. Bayro-Corrochano, E., Daniilidis, K., & Sommer, G. (1997). Hand-eye calibration in terms of motions of lines using geometric algebra. In *10th Scandinavian Conference on Image Analysis* (Vol. I, pp. 397–404). Finland: Lappeenranta.
13. Daniilidis, K., & Bayro-Corrochano, E. (1996). The dual quaternion approach to hand–eye calibration. In *IEEE Proceedings of the International Conference on Pattern Recognition (ICPR'96)* (Vol. I, pp. 318–322). Vienna, Austria.
14. Maybeck, P. (1979). *Stochastic models, estimation and control* (Vol. 141-3). New York: Academic Press
15. Kunze, S. (1999). Ein Hand–Auge–System zur visuell basierten Lokalisierung und Identifikation von Objekten. Diploma Thesis, Christian–Albrechts–Universität zu Kiel, Institut für Informatik und Praktische Mathematik,
16. Bayro-Corrochano, E., & Zhang, Y. (2000). The motor extended Kalman filter: A geometric approach for 3D rigid motion estimation. *Journal of Mathematical Imaging and Vision*, *13*(3), 205–227.

Chapter 16
Tracker Endoscope Calibration and Body-Sensor Calibration

In general, when the sensors are mounted on a robot arm, one can use the hand–eye calibration algorithm to calibrate them. In this chapter, we present the calibration of an endoscopic camera with respect to a tracking system and the case of a mobile robot for that one has to calibrate the robot's sensors with respect to the robot's global coordinate system.

16.1 Device-Camera Calibration

This section presents an algorithm in the conformal geometric algebra framework that computes the transformation (rotation and translation) relating the coordinate system of an endoscopic camera with the coordinate system of the markers placed on it. Such markers are placed in order to track the camera's position in real time by using an optical tracking system. The problem is an adaptation of the so-called hand–eye calibration, well known in robotics; however, we call it the *tracker endoscope* calibration. In this way, we can relate the preoperative data (3D model and planning of the surgery), with the data acquired in real time during the surgery by means of the optical tracking system and the endoscope.

16.1.1 Rigid Body Motion in CGA

In CGA, rotations are represented by the *rotors*, which are defined as

$$\boldsymbol{R} = e^{\frac{1}{2}\boldsymbol{b}\theta} = \cos\frac{\theta}{2} + \boldsymbol{b}\sin\frac{\theta}{2}, \tag{16.1}$$

where $\boldsymbol{b}$ is the bivector dual to the rotation axis and θ is the rotation angle. The rotation of an entity is carried out by multiplying it by the left with the rotor $\boldsymbol{R}$, and

© Springer Nature Switzerland AG 2020

E. Bayro-Corrochano, *Geometric Algebra Applications Vol. II*,
https://doi.org/10.1007/978-3-030-34978-3_16

by the right for the reversion of the rotor $\boldsymbol{R}$: $\boldsymbol{X}' = \boldsymbol{R}\boldsymbol{X}\widetilde{\boldsymbol{R}}$. Translation is carried out by the so-called *translator*:

$$\boldsymbol{T} = e^{\frac{e_\infty \boldsymbol{t}}{2}} = 1 + \frac{e_\infty \boldsymbol{t}}{2}, \tag{16.2}$$

where $\boldsymbol{t} \in \langle G_3 \rangle_1$ is the translation vector. Translations are applied in a similar way to rotations: $\boldsymbol{X}' = \boldsymbol{T}\boldsymbol{X}\widetilde{\boldsymbol{T}}$.

To express rigid body transformations, rotors and translators are applied consecutively. The result is called *motor*:

$$\boldsymbol{M} = \boldsymbol{T}\boldsymbol{R}. \tag{16.3}$$

Such an operator is applied to any entity of any dimension by multiplying the entity by the operator from the left, and by the reverse of the operator from the right: $\boldsymbol{X}' = \boldsymbol{M}\boldsymbol{X}\widetilde{\boldsymbol{M}}$. The motor $\boldsymbol{M}$ is a special multivector of even grade. To see its components, let us carry out the multiplication of $\boldsymbol{R}$ and $\boldsymbol{T}$

$$\begin{aligned}
\boldsymbol{M} = \boldsymbol{T}\boldsymbol{R} &= \left(1 + \frac{1}{2} e_\infty \boldsymbol{t}\right)\left(\cos\left(\frac{\theta}{2}\right) + \boldsymbol{b}\sin\left(\frac{\theta}{2}\right)\right), \\
&= \cos\left(\frac{\theta}{2}\right) + \boldsymbol{b}\sin\left(\frac{\theta}{2}\right) + \frac{1}{2} e_\infty \left(\boldsymbol{t}\left(\cos\left(\frac{\theta}{2}\right) + \boldsymbol{b}\sin\left(\frac{\theta}{2}\right)\right)\right), \\
&= = \boldsymbol{R} + \boldsymbol{R}'.
\end{aligned} \tag{16.4}$$

Since the multiplication of a vector $t \in \langle G_3 \rangle_1$ by a bivector $\boldsymbol{b} \in \langle G_3 \rangle_2$ results in a multivector of the form $\lambda_1 e_1 + \lambda_2 e_2 + \lambda_3 e_3 + \lambda_4 e_{123}$, and since $\boldsymbol{t}\cos\left(\frac{\theta}{2}\right) \in \langle G_3 \rangle_1$, we can rewrite (16.4) as

$$\begin{aligned}
\boldsymbol{M} &= \cos\left(\frac{\theta}{2}\right) + \boldsymbol{b}\sin\left(\frac{\theta}{2}\right) + \frac{1}{2} e_\infty \left(\boldsymbol{t}\cos\left(\frac{\theta}{2}\right) + \boldsymbol{t}\boldsymbol{b}\sin\left(\frac{\theta}{2}\right)\right), \\
&= \cos\left(\frac{\theta}{2}\right) + \boldsymbol{b}\sin\left(\frac{\theta}{2}\right) + \frac{1}{2} e_\infty \left(t\cos\left(\frac{\theta}{2}\right) + \lambda_1 e_1 + \lambda_2 e_2 + \lambda_3 e_3 + \lambda_4 e_{123}\right), \\
&= \cos\left(\frac{\theta}{2}\right) + \boldsymbol{b}\sin\left(\frac{\theta}{2}\right) + e_\infty \left(t' + \lambda e_{123}\right), \\
\boldsymbol{M} &= \cos\left(\frac{\theta}{2}\right) + \boldsymbol{b}\sin\left(\frac{\theta}{2}\right) + e_\infty t' + \lambda e_{\infty 123},
\end{aligned} \tag{16.5}$$

where $\boldsymbol{t}' \in \langle G_3 \rangle_1$ and $\lambda = \frac{1}{2}\lambda_4$. Note that $e_\infty t'$ is a bivector with components $e_{\infty 1}, e_{\infty 2}, e_{\infty 3}$. If we take only the bivectorial parts of the motor $\boldsymbol{M}$, we obtain

$$\begin{aligned}
\langle \boldsymbol{M} \rangle_2 &= \langle \boldsymbol{R} \rangle_2 + \langle \boldsymbol{R}' \rangle_2, \\
&= \boldsymbol{m} + \boldsymbol{m}' = \sin\left(\frac{\theta}{2}\right)\boldsymbol{b} + e_\infty \boldsymbol{t}'.
\end{aligned} \tag{16.6}$$

Therefore, if we express the vector $\boldsymbol{t}'$ in terms of their dual bivector $\boldsymbol{t}' = \boldsymbol{t}''I_E$, we can rewrite (16.6) as

$$\langle \boldsymbol{M} \rangle_2 = \boldsymbol{b}' I_E + e_\infty \boldsymbol{t}'' I_E. \tag{16.7}$$

If we consider the representation of the lines

$$\boldsymbol{L} = (\boldsymbol{a} - \boldsymbol{b}) I_E + e_\infty (\boldsymbol{a} \wedge \boldsymbol{b}) I_E = \boldsymbol{n} I_E + e_\infty \boldsymbol{m} I_E, \tag{16.8}$$

we observe that the bivectorial part of the motor $\boldsymbol{M}$ is in fact a line and it corresponds to the screw axis in which the rotation and translation of the object are carried out.

16.1.2 Hand–Eye Calibration in CGA

The formulation for the hand–eye calibration problem given in Chap. 15 reads

$$\boldsymbol{M}_A \boldsymbol{M}_X = \boldsymbol{M}_X \boldsymbol{M}_B, \tag{16.9}$$

where $\boldsymbol{M}_A = \mathbf{A} + \boldsymbol{A}'$, $\boldsymbol{M}_B = \boldsymbol{B} + \boldsymbol{B}'$, and $\boldsymbol{M}_X = \boldsymbol{R} + \boldsymbol{R}'$ (Sect. 16.1.1). Chapter 15 shows that the angle and pitch of the gripper (endoscope in our case) are equal to the angle and pitch of the camera (they remain invariant under coordinate transformations, which is known as the screw congruence theorem); therefore, the problem is solved using only the lines defined by the motors

$$\begin{aligned} \boldsymbol{L}_A = \boldsymbol{a} + \boldsymbol{a}' &= \boldsymbol{M}_X \boldsymbol{L}_B \widetilde{\boldsymbol{M}}_X, \\ &= (\boldsymbol{R} + \boldsymbol{R}')(\boldsymbol{b} + \boldsymbol{b}')\widetilde{(\boldsymbol{R} + \boldsymbol{R}')}, \\ &= \boldsymbol{R}\boldsymbol{b}\widetilde{\boldsymbol{R}} + e_\infty(\boldsymbol{R}\boldsymbol{b}\widetilde{\boldsymbol{R}}' + \boldsymbol{R}\boldsymbol{b}'\widetilde{\boldsymbol{R}} + \boldsymbol{R}'\boldsymbol{b}\widetilde{\boldsymbol{R}}, \end{aligned} \tag{16.10}$$

where $\boldsymbol{a}, \boldsymbol{a}', \boldsymbol{b}, \boldsymbol{b}'$ are bivectors (as in Eq. (16.6)). By separating the real part and the part multiplied by e_∞, we have

$$\boldsymbol{a} = \boldsymbol{R}\boldsymbol{b}\widetilde{\boldsymbol{R}}, \tag{16.11}$$

$$\boldsymbol{a}' = \boldsymbol{R}\boldsymbol{b}\widetilde{\boldsymbol{R}}' + \boldsymbol{R}\boldsymbol{b}'\widetilde{\boldsymbol{R}} + \boldsymbol{R}'\boldsymbol{b}\widetilde{\boldsymbol{R}}. \tag{16.12}$$

Multiplying from the right by $\boldsymbol{R}$ and using the relationship $\widetilde{\boldsymbol{R}}\boldsymbol{R}' + \widetilde{\boldsymbol{R}}'\boldsymbol{R} = 0$, the following relationships are obtained:

$$\boldsymbol{a}\boldsymbol{R} - \boldsymbol{R}\boldsymbol{b} = 0, \tag{16.13}$$

$$(\boldsymbol{a}'\boldsymbol{R} - \boldsymbol{R}\boldsymbol{b}') + (\boldsymbol{a}\boldsymbol{R}' - \boldsymbol{R}'\boldsymbol{b}) = 0, \tag{16.14}$$

which can be expressed in matrix form as

$$\begin{bmatrix} \boldsymbol{a}-\boldsymbol{b} & [\boldsymbol{a}+\boldsymbol{b}]_\times & \mathbf{0}_{3\times 1} & \mathbf{0}_{3\times 3} \\ \boldsymbol{a}'-\boldsymbol{b}' & [\boldsymbol{a}'+\boldsymbol{b}']_\times & \boldsymbol{a}-\boldsymbol{b} & [\boldsymbol{a}+\boldsymbol{b}]_\times \end{bmatrix} \begin{bmatrix} \boldsymbol{R} \\ \boldsymbol{R}' \end{bmatrix} = 0. \tag{16.15}$$

We call this 6×8 matrix D; the unknown vector $[\boldsymbol{R}, \boldsymbol{R}']^T$ is eight-dimensional. The notation $[u]_\times$ represents the skew-symmetric matrix formed with the vector u, which is given by

$$\hat{u} = \begin{bmatrix} 0 & -u_3 & u_2 \\ u_3 & 0 & -u_1 \\ -u_2 & u_1 & 0 \end{bmatrix}. \tag{16.16}$$

The matrix D is composed only of bivectors (blades of any other grade are not included), and therefore we can use the SVD method to find $[\boldsymbol{R} \;\; \boldsymbol{R}']^T$ as the kernel of D.

Considering that we have $n \geq 2$ movements, the following matrix is built:

$$C = \begin{bmatrix} D_1^T & D_2^T & D_3^T & D_4^T & \cdots \end{bmatrix}^T \tag{16.17}$$

in order to apply the SVD method and to find the solution for $[\boldsymbol{R}, \boldsymbol{R}']^T$. Since the range of the matrix C is at most 6, the last right two singular vectors, v_7 and v_8, correspond to the two singular values whose value is zero or near zero, and such vectors expand the null space of C. Therefore, as $[\boldsymbol{R}, \boldsymbol{R}']^T$ is a null vector of C, we can express it as a linear combination of v_7 and v_8. If we express these vectors in terms of two vectors of 4D $v_7 = (u_1, v_1)^T$ and $v_8 = (u_2, v_2)^T$, this linear combination can be expressed as

$$\begin{bmatrix} \boldsymbol{R} \\ \boldsymbol{R}' \end{bmatrix} = \alpha \begin{bmatrix} u_1 \\ v_1' \end{bmatrix} + \beta \begin{bmatrix} u_2 \\ v_2' \end{bmatrix}. \tag{16.18}$$

Taking into account the constraints

$$\boldsymbol{R}\widetilde{\boldsymbol{R}} = 1 \quad \text{and} \quad \widetilde{\boldsymbol{R}}\boldsymbol{R}' + \widetilde{\boldsymbol{R}}'\boldsymbol{R} = 0, \tag{16.19}$$

we obtain the following quadratic equations in terms of α and β:

$$\alpha^2 u_1^T u_1 + 2\alpha\beta u_1^T u_2 + \beta^2 u_2^T u_2 = 1, \tag{16.20}$$

$$\alpha^2 u_1^T v_1 + \alpha\beta(u_1^T v_2 + u_2^T v_1) + \beta^2 u_2^T v_2 = 0. \tag{16.21}$$

In order to solve these equations, we make a change of variable, substituting in (16.21) $\mu = \alpha/\beta$, and we obtain two solutions for μ. Going back to (16.20) and replacing the relationship $\alpha = \mu\beta$, we obtain

$$\beta^2(\mu^2 u_1^T u_1 + \mu(2u_1^T u_2) + u_2^T u_2 = 1, \tag{16.22}$$

which takes two solutions for β.

16.1.3 Tracker Endoscope Calibration

The surgery scenario is shown in Fig. 16.1, where the reader can see that there is a (rigid) transformation between the calibration grid and the Polaris system, $\boldsymbol{M}_{B_g}$. Such transformation will be used to validate the results of the endoscope-tracking system calibration method. The transformations involved in the problem are expressed as motors of the CGA: $\boldsymbol{M} = \boldsymbol{TR}$. They and the coordinate systems involved are as follows:

- $\mathbf{O_p}$: Reference frame of the Polaris system.
- $\mathbf{O_{m_e}}$: Reference frame of the spherical markers.
- $\mathbf{O_{c_e}}$: Reference frame of the endoscopic camera.
- $\mathbf{O_{g_c}}$: Reference frame of the calibration grid.
- $\boldsymbol{M}_{A_i}$: Transformation from $\mathbf{O_{g_c}}$ to $\mathbf{O_{c_e}}$ in the i-th movement of the camera. This transformation is given by the extrinsic parameters of the camera [1].
- $\boldsymbol{M}_{B_i}$: Transformation from $\mathbf{O_{m_e}}$ to $\mathbf{O_p}$. It is obtained by the optical tracking system.
- $\boldsymbol{M}_X$: Transformation from $\mathbf{O_{c_e}}$ to $\mathbf{O_{m_e}}$, obtained with the endoscope-tracking calibration algorithm.
- $\boldsymbol{M}_{B_g}$: Transformation from $\mathbf{O_{g_c}}$ to $\mathbf{O_p}$. It is obtained directly from the Polaris system by placing spherical markers on the plane where the calibration grid lies on.

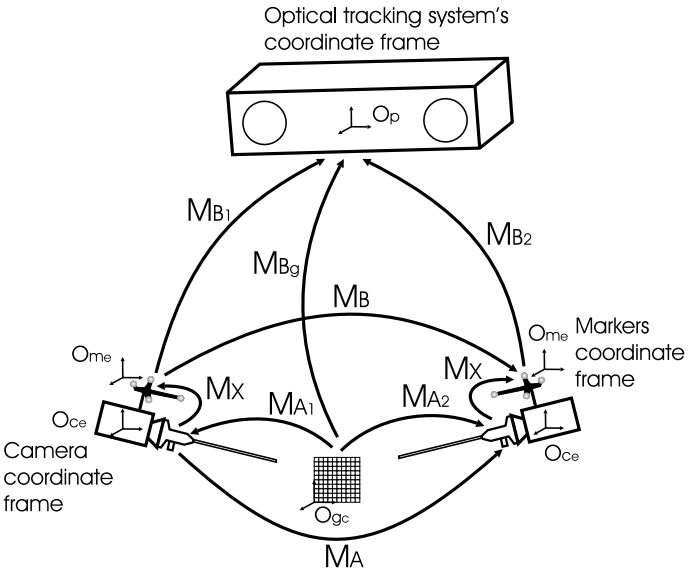


Fig. 16.1 The problem of the calibration between the endoscopic camera and the optical tracking system (Polaris)

The procedure is summarized as follows:

1. Given n movements of the endoscopic camera (we move it freely by hand to arbitrary positions), $\boldsymbol{M}_{B_i}$, and their corresponding movements $\boldsymbol{M}_{A_i}$, verify if their scalar parts are equal (screw congruence theorem with motors).
2. For the movements that fulfill the previous requirement, extract the directions and moments of the lines $\boldsymbol{L}_{A_i}$ and $\boldsymbol{L}_{B_i}$ defined by the motors. Build the matrix C as in (16.17).
3. Apply SVD to matrix C. Take the right singular vectors v_7 and v_8 corresponding to the two singular values nearest to zero (a threshold is applied by the noise).
4. Compute the coefficients for (16.21) and find the two solutions of μ.
5. For both values of μ, compute the value of $\mu^2 u_1^T u_1 + 2\mu u_1^T u_2 + u_2^T u_2$ and choose the one that gives bigger value. Then, compute α and β. The final solution is $\alpha v_7 + \beta v_8$.

To validate the accuracy of the estimated transformation $\boldsymbol{M}_X$, we use the calibration grid used to calibrate the endoscopic camera by Zhang's method [1]. Let $\boldsymbol{X}_g$ be the set of points corresponding to the corners of the calibration grid, referred to the coordinate system $\mathbf{O_{g_c}}$. These coordinates are expressed in millimeters, according to the size of each square in the calibration grid, whose sides are 1.25 mm in length in our case. Such points are projected to the camera following two paths: the first one is given applying $\boldsymbol{M}_{A_i}$ and then the matrix of the intrinsic parameters K, resulting in points x_{A_i}; the second one is obtained applying the transformations $\boldsymbol{M}_{B_g}$, $\widetilde{\boldsymbol{M}}_{B_i}$, and $\widetilde{\boldsymbol{M}}_X$, resulting in points x_{B_i}. A linear correction is applied to these points $x'_{B_i} = x_{B_i} + (c_{A_i} - c_{B_i})$, where c is the centroid of each point set. The transformation M_X was obtained with the exposed method.

1. Taking the points X_g in the grid reference frame, apply the transformation $\boldsymbol{M}_{A_i}$ to express them in the camera's reference frame. Let X_{A_i} be the resulting points.
2. Project the points X_{A_i} to the image plane using

$$x_{A_i} = K \ [\boldsymbol{R}_{M_A} \ t_{M_A}] \ X_{A_i}. \tag{16.23}$$

   These points should be projected on the corners of the squares in the calibration grid on the image.
3. Taking the points X_g, apply the transformations $\boldsymbol{M}_{B_g}$, $\widetilde{\boldsymbol{M}}_{B_i}$, and $\widetilde{\boldsymbol{M}}_X$. Let $X_{M_{B_i X}}$ be the resulting points.
4. Project the points $X_{M_{B_i X}}$ onto the image plane using

$$x_{B_i} = K \ [\boldsymbol{R}_{M_{B_i X}} \ t_{M_{B_i X}}] \ X_{M_{B_i X}}. \tag{16.24}$$

   In the ideal case (without noise), the projected points x_{A_i} should match the projected points x_{B_i}. However, as a result of noise in the Polaris readings or noise in the estimation of transformations, a small linear displacement between x_{A_i} and x_{B_i} is possible (see Fig. 16.2a). We can measure the error ϵ between the two projections as

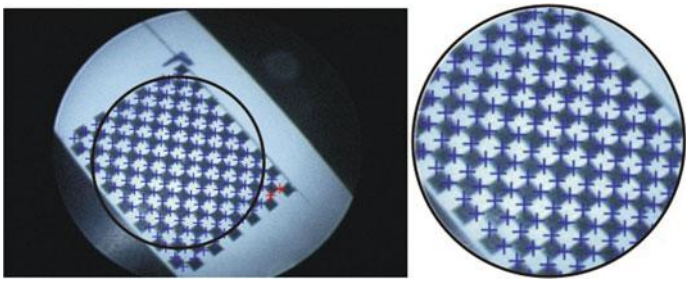


Fig. 16.2 Result of the projection and linear correction

$$\epsilon = \frac{\sum_{i=1}^{n} (x_{A_i} - x_{B_i})}{n}. \tag{16.25}$$

5. In order to correct the displacement, the centroid of each point set is calculated: c_{A_i} and c_{B_i}. Then, the points x_{B_i} are displaced in such a way that the centroids match:

$$x'_{B_i} = x_{B_i} + (c_{A_i} - c_{B_i}). \tag{16.26}$$

   After the displacement, the average error is calculated as

$$\epsilon' = \frac{\sum_{i=1}^{n} (x_{A_i} - x'_{B_i})}{n}. \tag{16.27}$$

   Figure 16.2b shows the result after doing the correction to the points shown in Fig. 16.2a.

The experiments were carried out using an endoscope from Karl Storz Endoscopy, which consists of a TeleCam SL II camera and an endoscope with a view angle of zero degrees. The movements were done by hand, to arbitrary positions in space. We test the algorithm computing the transformation $\boldsymbol{M}_X$ using a different number of motions each time. To measure the error between the projection, we take the Euclidean distance between each point x_{A_i} and its corresponding point x'_{B_i}: $Error = |x_{A_i} - x'_{B_i}|$. This error is a distance measured in pixels. In each motion, the image of the grid contains 154 points approximately.

16.2 Body-Sensor Calibration

The body–eye calibration problem is closely related to the hand–eye calibration, where in the latter we proceed as follows: given rigid displacements between the movements of a robot arm and the movements measured by a camera mounted on the arm, the unknown rigid transformation between the coordinate system of the arm

and of the cameras has to be computed. As it is shown in Chap. 15, this unknown transformation remains fixed through all the movements.

The use of the homogeneous transformation matrices is the usual way to formulate the hand–eye calibration problem. We denote by $\mathbf{X}$ the transformation from the coordinate system of the camera that of the gripper, by $\mathbf{B}_i$ the transformation matrix from the robot base to the gripper, and by $\boldsymbol{a}_i$ the transformation matrix that describes the rigid motions from the camera from one position to another. The well-known equation

$$\mathbf{AX} = \mathbf{XB} \tag{16.28}$$

was solved in a linear manner using dual quaternions or motors in [2, 3], where $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{X}$ are homogeneous transformation matrices of the form

$$\begin{bmatrix} \mathbf{R} & \mathbf{t} \\ 0^T & 1 \end{bmatrix}. \tag{16.29}$$

For the case where the sensors are mounted on a mobile robot, one extends in a certain way the hand–eye calibration method to multiple sensors. Thus, from now on we will call this kind of problem body–eye calibration.

16.2.1 Body–Eye Calibration

In the body–eye calibration, there are not many degrees of freedom as in the case of a robotic arm; furthermore, the rotation axes of the mobile robot are always parallel, which help us to simplify the problem. To solve the problem, we use motors of the conformal geometric algebra; this representation allows us to solve the problem straightforwardly and linearly. An abstraction of the setup is given in Fig. 16.3.

Next we will show the process to find the unknown transformation. First we rewrite (16.28) for the body–eye problem in terms of motors:

$$\boldsymbol{M}_A\boldsymbol{M} = \boldsymbol{M}\boldsymbol{M}_B, \tag{16.30}$$

or

$$\boldsymbol{M}_A = \boldsymbol{M}\boldsymbol{M}_B\widetilde{\boldsymbol{M}}. \tag{16.31}$$

This equation can be rewritten in terms of rotors and translators as follows:

$$\boldsymbol{M} = \boldsymbol{T}\boldsymbol{R} = \left(1 + \frac{1}{2}e_\infty \boldsymbol{t}\right)\boldsymbol{R} = \boldsymbol{R} + \frac{1}{2}e_\infty \boldsymbol{t}\boldsymbol{R} = \boldsymbol{R} + e_\infty \boldsymbol{r}, \tag{16.32}$$

where $\frac{1}{2}\boldsymbol{t}\boldsymbol{R}$ is equal to

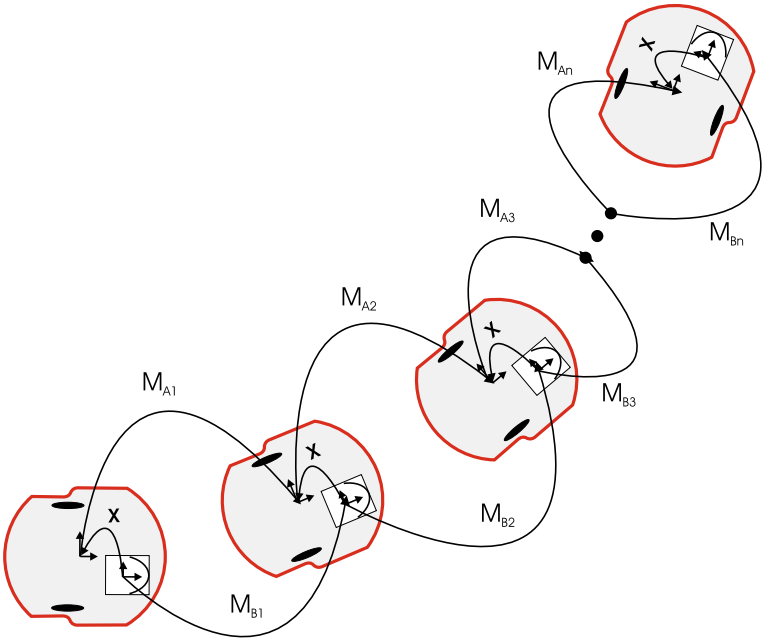


Fig. 16.3 Robot movements in order to calibrate a sensor mounted on it

$$
\begin{aligned}
\frac{1}{2}\boldsymbol{t}\boldsymbol{R} &= \frac{1}{2}\left(\alpha_1 e_1 + \alpha_2 e_2\right)\left(\cos\left(\frac{\theta}{2}\right) - \sin\left(\frac{\theta}{2}\right) e_{12}\right), \\
&= \frac{1}{2}\left(\alpha_1 e_1 + \alpha_2 e_2\right)\cos\left(\frac{\theta}{2}\right) - \frac{1}{2}\left(\alpha_1 e_1 + \alpha_2 e_2\right)\sin\left(\frac{\theta}{2}\right) e_{12}, \\
&= \frac{1}{2}\left[\left(\alpha_1 e_1 + \alpha_2 e_2\right)\cos\left(\frac{\theta}{2}\right) - \left(\alpha_1 e_2 - \alpha_2 e_1\right)\sin\left(\frac{\theta}{2}\right)\right], \\
&= \frac{1}{2}\left[\left(\alpha_1 \cos\left(\frac{\theta}{2}\right) + \alpha_2 \sin\left(\frac{\theta}{2}\right)\right) e_1 - \left(\alpha_2 \cos\left(\frac{\theta}{2}\right) - \alpha_1 \sin\left(\frac{\theta}{2}\right)\right) e_2\right].
\end{aligned}
$$

Note that the product $\frac{1}{2}\boldsymbol{t}\boldsymbol{R}$ is a vector, which is represented by $\boldsymbol{r}$ ($\boldsymbol{r} = \frac{1}{2}\boldsymbol{t}\boldsymbol{R}$). The motor can also be written as

$$
\boldsymbol{M} = \boldsymbol{R} + e_\infty \boldsymbol{r} = \cos\left(\frac{\theta}{2}\right) - \sin\left(\frac{\theta}{2}\right) e_{12} + e_\infty \boldsymbol{r} = \cos\left(\frac{\theta}{2}\right) + \boldsymbol{L}, \tag{16.33}
$$

where the line $\boldsymbol{L} = -\sin\left(\frac{\theta}{2}\right) e_{12} + e_\infty$ represents the screw axis of the motor (Fig. 16.4).

The inverse of the motor can be computed with

$$
\widetilde{\boldsymbol{M}} = \widetilde{\boldsymbol{T}}\widetilde{\boldsymbol{R}} = \widetilde{(\boldsymbol{R} + e_\infty \boldsymbol{r})} = \widetilde{\boldsymbol{R}} + \widetilde{e_\infty \boldsymbol{r}} = \widetilde{\boldsymbol{R}} - e_\infty \boldsymbol{r}. \tag{16.34}
$$

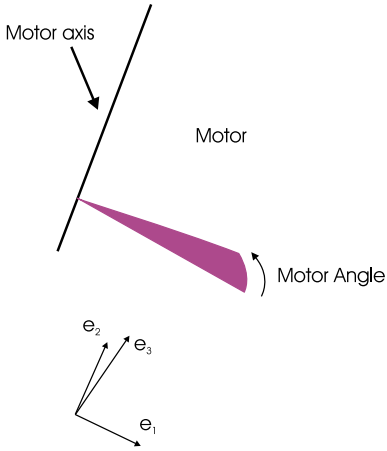


Fig. 16.4 Motor in the space and its correspondence axis line

The motor $\boldsymbol{M}$ and its inverse $\widetilde{\boldsymbol{M}}$ satisfy the following identity:

$$\boldsymbol{M}\widetilde{\boldsymbol{M}} = \widetilde{\boldsymbol{M}}\boldsymbol{M} = 1, \tag{16.35}$$

which can be rewritten using (16.33) and (16.34) as

$$\begin{aligned} \boldsymbol{M}\widetilde{\boldsymbol{M}} &= (\boldsymbol{R} + e_\infty \boldsymbol{r})(\widetilde{\boldsymbol{R}} - e_\infty \boldsymbol{r}), \\ &= \boldsymbol{R}\widetilde{\boldsymbol{R}} - \boldsymbol{R}e_\infty \boldsymbol{r} + e_\infty \boldsymbol{r}\widetilde{\boldsymbol{R}}, \\ &= \boldsymbol{R}\widetilde{\boldsymbol{R}} + e_\infty(-\boldsymbol{R}\boldsymbol{r} + \boldsymbol{r}\widetilde{\boldsymbol{R}}). \end{aligned} \tag{16.36}$$

Since we know that $\boldsymbol{R}\widetilde{\boldsymbol{R}} = 1$, we get the following identity:

$$\boldsymbol{r}\widetilde{\boldsymbol{R}} - \boldsymbol{R}\boldsymbol{r} = \widetilde{\boldsymbol{R}}\boldsymbol{r} - \boldsymbol{r}\boldsymbol{R} = 0. \tag{16.37}$$

Now we can rewrite (16.31) as

$$\begin{aligned} \boldsymbol{R}_A + e_\infty \boldsymbol{r}_A &= (\boldsymbol{R} + e_\infty \boldsymbol{r})(\boldsymbol{R}_B + e_\infty \boldsymbol{r}_B)(\widetilde{\boldsymbol{R}} - e_\infty \boldsymbol{r}), \\ &= \boldsymbol{R}\boldsymbol{R}_B\widetilde{\boldsymbol{R}} + \boldsymbol{R}e_\infty \boldsymbol{r}_B\widetilde{\boldsymbol{R}} + e_\infty \boldsymbol{r}\boldsymbol{R}_B\widetilde{\boldsymbol{R}} - \boldsymbol{R}\boldsymbol{R}_B e_\infty \boldsymbol{r}, \end{aligned} \tag{16.38}$$

from which we get

$$\boldsymbol{R}_A = \boldsymbol{R}\boldsymbol{R}_B\widetilde{\boldsymbol{R}}, \tag{16.39}$$

which is equal to

$$\cos\left(\frac{\theta_A}{2}\right) + \sin\left(\frac{\theta_A}{2}\right)\boldsymbol{n}_A = \boldsymbol{R}\left(\cos\left(\frac{\theta_B}{2}\right) + \sin\left(\frac{\theta_B}{2}\right)n_B\right)\widetilde{\boldsymbol{R}},$$

$$
\begin{aligned}
&= \cos\left(\frac{\theta_B}{2}\right) \boldsymbol{R}\widetilde{\boldsymbol{R}} + \sin\left(\frac{\theta_B}{2}\right) \boldsymbol{R}\boldsymbol{n}_B\widetilde{\boldsymbol{R}}, \\
&= \cos\left(\frac{\theta_B}{2}\right) + \sin\left(\frac{\theta_B}{2}\right) \boldsymbol{R}\boldsymbol{n}_B\widetilde{\boldsymbol{R}}, \\
&= \cos\left(\frac{\theta_B}{2}\right) + \sin\left(\frac{\theta_B}{2}\right) \boldsymbol{n}_B. \qquad (16.40)
\end{aligned}
$$

Equating the scalar and bivector parts of both sides of the equations, we get the following equations:

$$
\begin{aligned}
\cos\left(\frac{\theta_A}{2}\right) &= \cos\left(\frac{\theta_B}{2}\right), \\
\sin\left(\frac{\theta_A}{2}\right) \boldsymbol{n}_A &= \sin\left(\frac{\theta_B}{2}\right) \boldsymbol{n}_B. \qquad (16.41)
\end{aligned}
$$

16.2.2 Algorithm Simplification

Taking this fact into account, we can be sure that only the bivector parts of $\boldsymbol{M}_A$ and $\boldsymbol{M}_B$ will contribute to the computation of the unknown $\boldsymbol{M}_X$. Using (16.33), we can define the body–eye calibration problem as

$$\boldsymbol{L}_A = \boldsymbol{M}\boldsymbol{L}_B\widetilde{\boldsymbol{M}}. \qquad (16.42)$$

Therefore, the angle and pitch of the motors $\boldsymbol{M}_A$ and $\boldsymbol{M}_B$ are always equal through all the hand movements. Thus, it suffices to regard the rotation axes $(\boldsymbol{L}_A, \boldsymbol{L}_B)$ of the involved motors $(\boldsymbol{M}_A, \boldsymbol{M}_B)$. This is also known as the screw congruence theorem [4]. However, thanks to the use of conformal geometric algebra, the proof of this theorem was reduced to one step shown by (16.40). Since the body–eye problem is a matter of motion of lines, we define two of them as

$$
\begin{aligned}
\boldsymbol{L}_A &= \boldsymbol{n}_A + e_\infty \boldsymbol{m}_A, \\
\boldsymbol{L}_B &= \boldsymbol{n}_B + e_\infty \boldsymbol{m}_B. \qquad (16.43)
\end{aligned}
$$

Substituting the above equations in (16.42), we get

$$
\begin{aligned}
\boldsymbol{n}_A + e_\infty \boldsymbol{m}_A &= (\boldsymbol{R} + e_\infty \boldsymbol{r})(\boldsymbol{n}_B + e_\infty \boldsymbol{m}_B)(\widetilde{\boldsymbol{R}} - e_\infty \boldsymbol{r}), \\
&= (\boldsymbol{R}\boldsymbol{n}_B\widetilde{\boldsymbol{R}} + \boldsymbol{R}e_\infty \boldsymbol{m}_B\widetilde{\boldsymbol{R}} + e_\infty \boldsymbol{r}n_B\widetilde{\boldsymbol{R}} - \boldsymbol{R}\boldsymbol{n}_B e_\infty \boldsymbol{r}).
\end{aligned}
$$

Equating both sides of the equation, we get

$$\boldsymbol{n}_A = \boldsymbol{R}\boldsymbol{n}_B\widetilde{\boldsymbol{R}}, \tag{16.44}$$
$$\boldsymbol{m}_A = \boldsymbol{R}\boldsymbol{m}_B\widetilde{\boldsymbol{R}} + \boldsymbol{r}n_B\widetilde{\boldsymbol{R}} - \boldsymbol{R}\boldsymbol{n}_B\boldsymbol{r}. \tag{16.45}$$

Multiplying on the right by $\boldsymbol{R}$ gives

$$\boldsymbol{n}_A\boldsymbol{R} = \boldsymbol{R}\boldsymbol{n}_B, \tag{16.46}$$
$$\boldsymbol{m}_A\boldsymbol{R} = \boldsymbol{R}\boldsymbol{m}_B + \boldsymbol{r}\boldsymbol{n}_B - \boldsymbol{R}\boldsymbol{n}_B\boldsymbol{r}\boldsymbol{R}. \tag{16.47}$$

Using (16.37) and (16.46), we get

$$\boldsymbol{m}_A\boldsymbol{R} = \boldsymbol{R}\boldsymbol{m}_B + \boldsymbol{r}\boldsymbol{n}_B - \boldsymbol{n}_A\boldsymbol{r}, \tag{16.48}$$

and thus we can write

$$\boldsymbol{n}_A\boldsymbol{R} - \boldsymbol{R}\boldsymbol{n}_B = 0, \tag{16.49}$$
$$(\boldsymbol{m}_A\boldsymbol{R} - \boldsymbol{R}\boldsymbol{m}_B) + (\boldsymbol{n}_A\boldsymbol{r} - \boldsymbol{r}\boldsymbol{n}_B) = 0. \tag{16.50}$$

Equation (16.49) can be written as

$$\boldsymbol{n}_A\boldsymbol{R} - \boldsymbol{n}_B\boldsymbol{R} = (n_A - n_B)\boldsymbol{R} = 0. \tag{16.51}$$

Since in this case the rotation axes are parallel we have that $\boldsymbol{n}_B$ and $\boldsymbol{n}_A$ are equal; thus, the equation is zero. Therefore, only (16.50) is useful when solving the problem. Equation (16.50) can be written as a matrix vector equation:

$$\begin{bmatrix} \alpha_2 - \alpha_1 & -\beta_1 - \beta_2 & 0 & 2 \\ \beta_2 - \beta_1 & \alpha_2 + \alpha_1 & -2 & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{R} \\ \boldsymbol{r} \end{bmatrix} = 0. \tag{16.52}$$

This will be written as

$$A\mathbf{x} = 0. \tag{16.53}$$

In (16.52), the scalars α_1, β_1 and α_2, β_2 represent the components of the moments $\boldsymbol{m}_B$ and $\boldsymbol{m}_A$, respectively, that is,

$$\boldsymbol{m}_B = \alpha_1 e_1 + \beta_1 e_2, \tag{16.54}$$
$$\boldsymbol{m}_A = \alpha_2 e_1 + \beta_2 e_2. \tag{16.55}$$

Note that the matrix has two equations; thus, we need at least two motions to compute the four unknowns. Stacking couples of rows for more than two motions, we get an overdetermined equation system that can be solved using SVD. That is, we compute the SVD of $A = USV^T$, and then we choose the singular vector with the smallest singular value as the solution. Let $v = \alpha_1 e_1 + \alpha_2 e_2 + \alpha_3 e_3 + \alpha_4 e_4$ be such a vector, then we define a motor as

$$\boldsymbol{M}_0 = \alpha_1 - \alpha_2 e_{12} + e_\infty(\alpha_3 e_1 + \alpha_4 e_2). \tag{16.56}$$

Normalizing this motor, we get

$$\boldsymbol{M} = \frac{\boldsymbol{M}_0}{\sqrt{\boldsymbol{M}_0 \widetilde{\boldsymbol{M}}_0}}, \tag{16.57}$$

which is the solution motor. The calibration algorithm is defined as follows:

1. Carry out several movements with the robot, (Fig. 16.3).
2. For each movement, build the motor $\boldsymbol{M}_A = \boldsymbol{T}_A \boldsymbol{R}_A$, representing the robot movement. The parameters of the motor are estimated using odometry.
3. For each motor $\boldsymbol{M}_A$, build a correspondence motor $\boldsymbol{M}_B = \boldsymbol{T}_B \boldsymbol{R}_B$, representing the sensor motion.
4. Extract the line representing the axis of the motor $(\boldsymbol{L}_A, \boldsymbol{L}_B)$ for each pairs of motors $\boldsymbol{M}_A$ and $\boldsymbol{M}_B$, this can be done with $\boldsymbol{L}_A = \langle \boldsymbol{M}_A \rangle_2$ and $\boldsymbol{L}_B = \langle \boldsymbol{M}_B \rangle_2$ (Fig. 16.4).
5. Using the moments of the lines $\boldsymbol{L}_A$ and $\boldsymbol{L}_B$, $\boldsymbol{m}_B = \alpha_1 e_1 + \beta_1 e_2$ and $\boldsymbol{m}_A = \alpha_2 e_1 + \beta_2 e_2$, add two rows to the matrix A.
6. For each movement of the robot, go to step 2.
7. Compute the SVD of the matrix A, and then choose the vector with the smallest singular value as the solution. Let $v = \alpha_1 e_1 + \alpha_2 e_2 + \alpha_3 e_3 + \alpha_4 e_4$ be such a vector. Then we define the following motor: $\boldsymbol{M}_0 = \alpha_1 - \alpha_2 e_{12} + e_\infty(\alpha_3 e_1 + \alpha_4 e_2)$.
8. Finally, the solution is found normalizing the motor $\boldsymbol{M}_0$, that is, $\boldsymbol{M} = \frac{\boldsymbol{M}_0}{\sqrt{\boldsymbol{M}_0 \widetilde{\boldsymbol{M}}_0}}$.

16.3 Conclusions

Currently, the use of endoscopes in surgeries is a common task. It provides the surgeon with very useful information, but to achieve that, camera calibration is needed. Additionally, if it is necessary to display the endoscopic image in a special frame, relating it with a virtual model of the patient, then a second calibration procedure is essential. Such a procedure finds the transformation between the camera and the tracking system used. That is the problem we deal in the first part of the chapter, which is addressed as a hand–eye calibration problem in the geometric algebra framework.

In general, when the sensors are mounted on a robot arm, one can use the hand–eye calibration algorithm to calibrate them. However, for the case of a mobile robot, one has to calibrate the sensors with respect to the robot's global coordinate system, so that the robot would be able to fuse the data of different sensors. The second part of the chapter presents a calibration algorithm for the standard sensors mounted on a mobile robot. Once the sensors are calibrated with respect to the robot platform, the robot can autonomously build 3D maps, navigate, and relocalize itself. These tasks will be presented in Chap. 18.

References

1. Zhang, Z. (1992). Iterative point matching for registration of free-form curves. Technical report 1658, INRIA.
2. Daniilidis, K., & Bayro-Corrochano, E. (1996). The dual quaternion approach to hand–eye calibration. In *IEEE Proceedings of the International Conference on Pattern Recognition (ICPR'96)* (Vol. I, pp. 318–322), Vienna, Austria.
3. Bayro-Corrochano, E., Daniilidis, K., & Sommer, G. (1997). Hand-eye calibration in terms of motions of lines using geometric algebra. In *10th Scandinavian Conference on Image Analysis* (Vol. I, pp. 397–404). Finland: Lappeenranta.
4. Chen, H. (1991). A screw motion approach to uniqueness analysis of head–eye geometry. In *IEEE Conference on Computer Vision and Pattern Recognition*, Maui, Hawaii, 3–6 June (pp. 145–151).

Chapter 17
Tracking, Grasping, and Object Manipulation

In this chapter, we utilize the conformal geometric algebra for the development of concepts and computer algorithms in the domain of robot vision. We present an interesting application of fussy logic and conformal geometric algebra for grasping using the Barret Hand. We present real-time algorithms for a real scenario of perception, approach, and action that handles a number of real grasping and manipulation tasks.

17.1 Tracking

First we will show an example using our formulation of the Jacobian. This is the control of a binocular robot head based on a pan–tilt unit (Fig. 17.1). In this head there are just two axes of movement, but the position of the focus of attention has three degrees of freedom related to the joint angles by using Eq. (6.131) of direct kinematics. Now the axis lines are needed and they can be computed from the extrinsic parameters of the camera calibration. In this example, the axis lines have the following equations:

$$\boldsymbol{L}_1 = -e_{31}, \tag{17.1}$$
$$\boldsymbol{L}_2 = e_{12} + d_1 e_1 e_\infty, \tag{17.2}$$
$$\boldsymbol{L}_3 = e_1 e_\infty, \tag{17.3}$$

where d_1 is the height of the tilt axis to the coordinate system. Note that $\boldsymbol{L}_1$ crosses the origin and its direction is e_2. Also, the axis $\boldsymbol{L}_3$ is a line at infinity that represents the translation of the focus of attention in front of the cameras.

Since $\boldsymbol{M}_i = e^{-\frac{1}{2}q_i \boldsymbol{L}_i}$ and $\widetilde{\boldsymbol{M}}_i = e^{\frac{1}{2}q_i \boldsymbol{L}_i}$, the position of the end-effectors is computed as

© Springer Nature Switzerland AG 2020

E. Bayro-Corrochano, *Geometric Algebra Applications Vol. II*,
https://doi.org/10.1007/978-3-030-34978-3_17

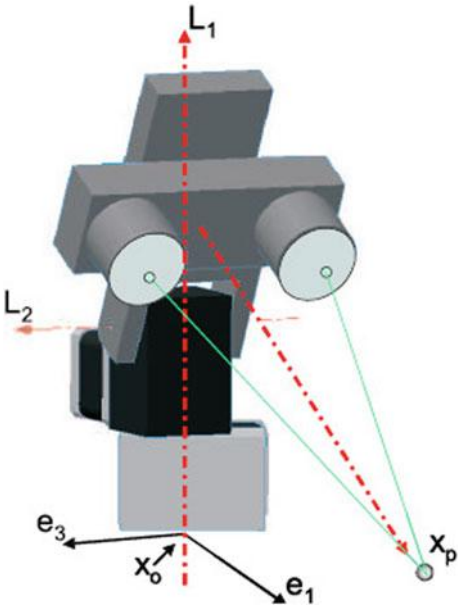


Fig. 17.1 Pan–tilt unit

$$\boldsymbol{x}'_p = \boldsymbol{M}_1\boldsymbol{M}_2\boldsymbol{M}_3\boldsymbol{x}_p\widetilde{\boldsymbol{M}}_3\widetilde{\boldsymbol{M}}_2\widetilde{\boldsymbol{M}}_1, \tag{17.4}$$

$$\boldsymbol{L}'_3 = \boldsymbol{M}_1\boldsymbol{M}_2\boldsymbol{L}_3\widetilde{\boldsymbol{M}}_2\widetilde{\boldsymbol{M}}_1, \tag{17.5}$$

$$\boldsymbol{L}'_2 = \boldsymbol{M}_1\boldsymbol{L}_2\widetilde{\boldsymbol{M}}_1, \tag{17.6}$$

$$\boldsymbol{L}'_1 = \boldsymbol{L}_1. \tag{17.7}$$

Now, using the equation of differential kinematics (5.98), the state variable representation of the system is written as follows:

$$\begin{cases} \dot{\boldsymbol{x}}'_p = \boldsymbol{x}' \cdot \begin{pmatrix} \boldsymbol{L}'_1 & \boldsymbol{L}'_2 & \boldsymbol{L}'_3 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} \\ \boldsymbol{y} = \boldsymbol{x}'_p, \end{cases} \tag{17.8}$$

where the position focus of attention $\boldsymbol{x}_p$ at $t = 0$ is given by the conformal mapping of $\boldsymbol{x}_{p_e} = d_3e_1 + (d_1 + d_2)e_2$, the line $\boldsymbol{L}'_i$ is the current position of $\boldsymbol{L}_i$, and u_i is the velocity of the joint i of the system.

17.1.1 Exact Linearization via Feedback

Now the following state feedback control law is chosen in order to get a new linear and controllable system. Figure 17.2 shows the plant

Fig. 17.2 Block diagram of the pan–tilt unit

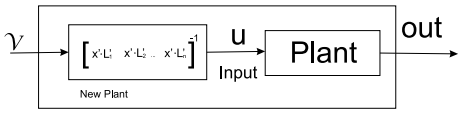


Fig. 17.3 Block diagram of the new system

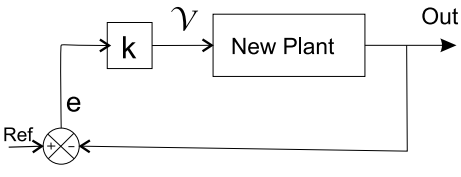


Fig. 17.4 Block diagram of closed-loop control

Figure 17.3 shows the new system, adding a block that linearizes the system as follows:

$$\begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} = \left(\boldsymbol{x}'_p \cdot \boldsymbol{L}'_1 \; \boldsymbol{x}'_p \cdot \boldsymbol{L}'_2 \; \boldsymbol{x}'_p \cdot \boldsymbol{L}'_3 \right)^{-1} \begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix}, \tag{17.9}$$

where $V = (v_1, v_2, v_3)^T$ is the new input to the linear systems, then we rewrite the equations of the system as follows:

$$\begin{cases} \dot{\boldsymbol{x}}'_p = V, \\ \boldsymbol{y} = \boldsymbol{x}'_p. \end{cases} \tag{17.10}$$

The problem of following a constant reference x_t is solved computing the error between end-effector position $\boldsymbol{x}'_p$ and the target position $\boldsymbol{x}_t$ as $e_r = (\boldsymbol{x}'_p \wedge \boldsymbol{x}_t) \cdot e_\infty$. The control law is then given as

$$V = -ke_r, \tag{17.11}$$

where k is the control gain (Fig. 17.4).

This error is small if the control system is accomplishing its task. It is mapped to an error in the joint space using the inverse Jacobian:

$$U = J^{-1}V. \tag{17.12}$$

Computing the Jacobian $J = \boldsymbol{x}'_p \cdot \left(\boldsymbol{L}'_1 \; \boldsymbol{L}'_2 \; \boldsymbol{L}'_3 \right)$

$$j_1 = \boldsymbol{x}'_p \cdot (\boldsymbol{L}_1), \tag{17.13}$$

$$j_2 = \boldsymbol{x}'_p \cdot (\boldsymbol{M}_1 \boldsymbol{L}_2 \widetilde{\boldsymbol{M}}_1), \tag{17.14}$$

$$j_3 = \boldsymbol{x}'_p \cdot (\boldsymbol{M}_1 \boldsymbol{M}_2 \boldsymbol{L}_3 \widetilde{\boldsymbol{M}}_2 \widetilde{\boldsymbol{M}}_1). \tag{17.15}$$

Once that we have the Jacobian, it is easy to compute the dq_i, using Crammer's rule:

$$\begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} = (j_1 \wedge j_2 \wedge j_3)^{-1} \cdot \begin{pmatrix} V \wedge j_2 \wedge j_3 \\ j_1 \wedge V \wedge j_3 \\ j_1 \wedge j_2 \wedge V \end{pmatrix}, \tag{17.16}$$

$$\begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} = \frac{\begin{pmatrix} ((\boldsymbol{x}'_p \wedge \boldsymbol{x}_t) \cdot e_\infty) \wedge (\boldsymbol{x}'_p \cdot \boldsymbol{L}'_2) \wedge (\boldsymbol{x}'_p \cdot \boldsymbol{L}'_3) \\ (\boldsymbol{x}'_p \cdot \boldsymbol{L}'_1) \wedge ((\boldsymbol{x}'_p \wedge \boldsymbol{x}_t) \cdot e_\infty) \wedge (\boldsymbol{x}'_p \cdot \boldsymbol{L}'_3) \\ (\boldsymbol{x}'_p \cdot \boldsymbol{L}'_1) \wedge (\boldsymbol{x}'_p \cdot \boldsymbol{L}'_2) \wedge ((\boldsymbol{x}'_p \wedge \boldsymbol{x}_t) \cdot e_\infty) \end{pmatrix}}{(\boldsymbol{x}'_p \cdot \boldsymbol{L}'_1) \wedge (\boldsymbol{x}'_p \cdot \boldsymbol{L}'_2) \wedge (\boldsymbol{x}'_p \cdot \boldsymbol{L}'_3)}. \tag{17.17}$$

This is possible because $j_1 \wedge j_2 \wedge j_3 = det(J) I_e$. Finally, we have dq_i, which will tend to reduce these errors.

17.1.2 Visual Jacobian

A point in the image is given by $s = (x, y)^T$, whereas a 3D point is represented as X. The relationship between $\dot{s}$ and $\dot{\boldsymbol{x}}$ is called visual Jacobian. Considering a camera in a general position, its projection matrix is represented by the planes π_1, π_2, and π_3 (see more details in [1]):

$$P = \begin{pmatrix} \pi_1 \\ \pi_2 \\ \pi_3 \end{pmatrix}. \tag{17.18}$$

The point X is projected in the image in the point:

$$s = \begin{pmatrix} \frac{\pi_1 \cdot X}{\pi_3 \cdot X} \\ \frac{\pi_2 \cdot X}{\pi_3 \cdot X} \end{pmatrix}. \tag{17.19}$$

To simplify the explanation, the $\boldsymbol{x}$-variable is introduced and its time derivative $\dot{\boldsymbol{x}}$ is defined as

$$\boldsymbol{x} = \begin{pmatrix} \pi_1 \cdot X \\ \pi_2 \cdot X \\ \pi_3 \cdot X \end{pmatrix}, \quad \dot{\boldsymbol{x}} = \begin{pmatrix} \pi_1 \cdot \dot{\boldsymbol{x}} \\ \pi_2 \cdot \dot{\boldsymbol{x}} \\ \pi_3 \cdot \dot{\boldsymbol{x}} \end{pmatrix}, \tag{17.20}$$

Now s is given by $s_1 = \frac{x_1}{x_3}$ and its derivative reads

$$\dot{s}_1 = \dot{\boldsymbol{x}}_1 \frac{1}{\boldsymbol{x}_3} + \boldsymbol{x}_1 \left(\frac{-\dot{\boldsymbol{x}}_3}{x_3^2} \right), \tag{17.21}$$

$$\dot{s}_1 = \frac{\boldsymbol{x}_3 \dot{\boldsymbol{x}}_1 - \boldsymbol{x}_1 \dot{\boldsymbol{x}}_3}{x_3^2}. \tag{17.22}$$

By substituting x and $\dot{\boldsymbol{x}}$ in Eq. 17.22, one obtains

$$\dot{s}_1 = \kappa \left[(\pi_3 \cdot X)\, \pi_1 - (\pi_1 \cdot X)\, \pi_3 \right] \cdot \dot{\boldsymbol{x}}, \tag{17.23}$$

$$\dot{s}_1 = \kappa \left[X \cdot (\pi_3 \wedge \pi_1) \right] \cdot \dot{\boldsymbol{x}}, \tag{17.24}$$

where $\kappa = \frac{1}{x_3^2}$. Carrying out the same steps for s_2, it is possible to write the equation

$$\dot{s} = \kappa X \cdot \begin{pmatrix} \pi_1 \wedge \pi_3 \\ \pi_2 \wedge \pi_3 \end{pmatrix} \cdot \dot{\boldsymbol{x}}. \tag{17.25}$$

Geometrically, $\pi_1 \wedge \pi_3$ represents a line of intersection of the planes π_1 and π_3. Denoting the intersection lines by $\boldsymbol{L}_x$ and $\boldsymbol{L}_y$

$$\boldsymbol{L}_x = \pi_1 \wedge \pi_3, \tag{17.26}$$

$$\boldsymbol{L}_y = \pi_2 \wedge \pi_3. \tag{17.27}$$

It is possible to rewrite Eq. 17.25 as

$$\dot{s} = \kappa X \cdot \begin{pmatrix} \boldsymbol{L}_x \\ \boldsymbol{L}_y \end{pmatrix} \cdot \dot{\boldsymbol{x}}. \tag{17.28}$$

In order to close the loop between perception and action, the relationship between velocities in the points of the image and the velocities in the joints of the pan–tilt unit is computed (see [2]).

Taking the equation of differential kinematics (17.8) and visual Jacobian (17.28), it is possible to write a new expression:

$$\dot{s} = \kappa \begin{pmatrix} \left(\boldsymbol{X}' \cdot \boldsymbol{L}_x'\right) \cdot \left(\boldsymbol{X}' \cdot \boldsymbol{L}_1'\right) & \left(\boldsymbol{X}' \cdot \boldsymbol{L}_x'\right) \cdot \left(\boldsymbol{X}' \cdot \boldsymbol{L}_2'\right) \\ \left(\boldsymbol{X}' \cdot \boldsymbol{L}_y'\right) \cdot \left(\boldsymbol{X}' \cdot \boldsymbol{L}_1'\right) & \left(\boldsymbol{X}' \cdot \boldsymbol{L}_y'\right) \cdot \left(\boldsymbol{X}' \cdot \boldsymbol{L}_2'\right) \end{pmatrix} \dot{q}. \tag{17.29}$$

We can write a similar expression using the differential kinematics of the Barrett hand. Equation (17.29) is very useful to design a control law to track an object or to grasp it.

17.1.3 Exact Linearization via Feedback

Now the following state feedback control law is chosen in order to get a new linear and controllable system:

$$\begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = \begin{pmatrix} (\boldsymbol{X}' \cdot \boldsymbol{L}'_x) \cdot (\boldsymbol{X}' \cdot \boldsymbol{L}'_1) & (\boldsymbol{X}' \cdot \boldsymbol{L}'_x) \cdot (\boldsymbol{X}' \cdot \boldsymbol{L}'_2) \\ (\boldsymbol{X}' \cdot \boldsymbol{L}'_y) \cdot (\boldsymbol{X}' \cdot \boldsymbol{L}'_1) & (\boldsymbol{X}' \cdot \boldsymbol{L}'_y) \cdot (\boldsymbol{X}' \cdot \boldsymbol{L}'_2) \end{pmatrix}^{-1} \begin{pmatrix} v_1 \\ v_2 \end{pmatrix}, \quad (17.30)$$

where $V = (v_1, v_2)^T$ is the new input to the linear system, then we rewrite the equations of the system:

$$\begin{cases} \dot{s}'_p = V, \\ \boldsymbol{y} = s'_p. \end{cases} \quad (17.31)$$

17.1.4 Experimental Results

In this experiment, the binocular head should smoothly track a target. Figure 17.5 shows the $3D$ coordinates of the focus of attention. Figure 17.6 shows examples of the image sequence. We can see that the curves of the $3D$ object trajectory are very rough. However, the control rule manages to keep the trajectory of the pan–tilt unit smooth.

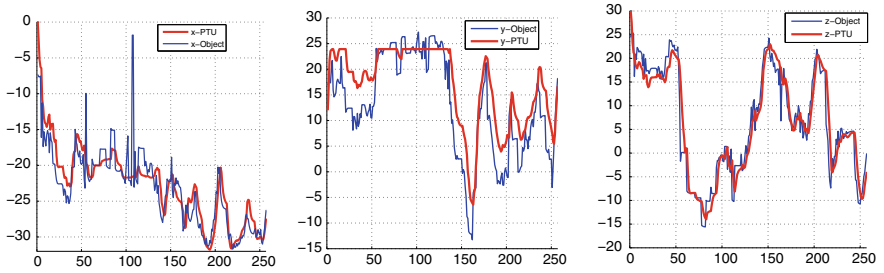


Fig. 17.5 x-, y-, and z-coordinates of the focus of attention

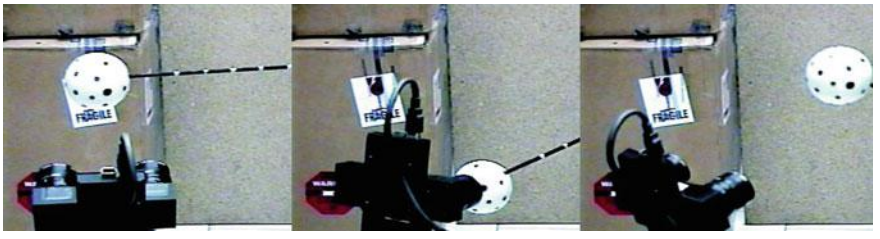


Fig. 17.6 Sequence of tracking

In this experiment, the coordinate system is in the center of the camera. Then the principal planes of the camera are given by

$$\pi_1 = f_x e_1 + x_o e_3, \tag{17.32}$$
$$\pi_2 = f_y e_2 + y_o e_3, \tag{17.33}$$
$$\pi_3 = e_3, \tag{17.34}$$

where f_x, f_y, x_o, and y_o are the camera's parameters. Using these planes, we compute the lines $\boldsymbol{L}_x$ and $\boldsymbol{L}_y$ using the known axis of the pan–tilt unit:

$$\boldsymbol{L}_1 = e_{23} + d_1 e_2, \tag{17.35}$$
$$\boldsymbol{L}_2 = e_{12} + d_2 e_2. \tag{17.36}$$

Note that the tilt axis is called $\boldsymbol{L}_1$ and the pan axis is $\boldsymbol{L}_2$, because the coordinate system is related to the camera. Also, $\boldsymbol{L}'_2$ is a function of the tilt angle $\boldsymbol{L}'_2 = \boldsymbol{M}_1 \boldsymbol{L}_2 \tilde{\boldsymbol{M}}_1$ with $\boldsymbol{M}_1 = \cos(\theta_{tilt}) + \sin(\theta_{tilt})\boldsymbol{L}_1$. In this experiment, a point on the circuit board was selected and tracked by using the *L*ucas–Kanade Tracking (LKT) algorithm. Its displacement in the image was transformed into velocities of the pan–tilt's joint, using the visual-mechanical Jacobian of Eq. (17.29).

As a result, we can see in Fig. 17.7 a sequence of pictures captured by the robot. In these images, the position of the board does not change while the background is changing continuously.

17.2 Barrett Hand Direct Kinematics

The direct kinematics involves the computation of the position and orientation of the end-effector given the parameters of the joints. Direct kinematics can be easily computed given the lines of the axes of screws.

In order to explain the kinematics of a Barrett hand, we show the kinematics of one finger. In this example, we will assume that the finger is totally extended. Note that such a hypothetical position is not reachable in normal operation, but this simplifies the explanation.

We start by denoting some points on the finger that help to describe the finger position:

$$x_{1o} = A_w e_1 + A_1 e_2 + D_w e_3, \tag{17.37}$$
$$x_{2o} = A_w e_1 + (A_1 + A2) e_2 + D_w e_3, \tag{17.38}$$
$$x_{3o} = A_w e_1 + (A_1 + A_2 + A_3) e_2 + D_w e_3. \tag{17.39}$$

The points x_{1o}, x_{2o}, and x_{3o} describe the position of each joint and the end of the finger in Euclidean space (see Fig. 17.8).

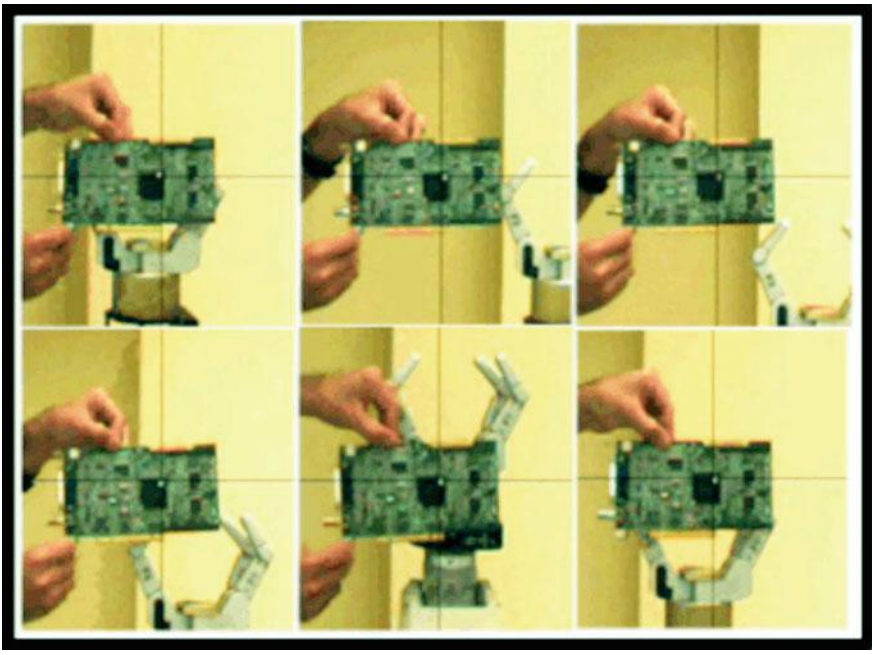


Fig. 17.7 Sequence of tracking

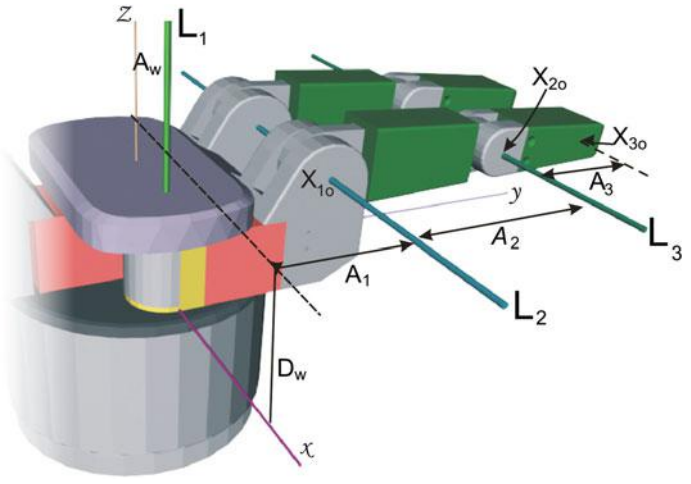


Fig. 17.8 Barrett hand in hypothetical position

Having defined these points, it is quite simple to calculate the axes that will be used as motor's axes:

$$\boldsymbol{L}_{1o} = -A_w(e_2 \wedge e_\infty) + e_{12}, \tag{17.40}$$
$$\boldsymbol{L}_{2o} = \left(\boldsymbol{x}_{c_{1o}} \wedge e_1 \wedge e_\infty\right) I_c, \tag{17.41}$$
$$\boldsymbol{L}_{3o} = \left(\boldsymbol{x}_{c_{2o}} \wedge e_1 \wedge e_\infty\right) I_c. \tag{17.42}$$

When the hand is initialized, the fingers move away to *home position*, which is $\Phi_2 = 2.46°$ in joint two and $\Phi_3 = 50°$ in joint three. In order to move the finger from this hypothetical position to its home position, the appropriate transformation needs to be computed as follows:

$$\boldsymbol{M}_{2o} = \cos(\Phi_2/2) - \sin(\Phi_2/2)\boldsymbol{L}_{2o}, \tag{17.43}$$
$$\boldsymbol{M}_{3o} = \cos(\Phi_3/2) - \sin(\Phi_3/2)\boldsymbol{L}_{3o}. \tag{17.44}$$

Having obtained these transformations, we then apply them to those points and lines that have to be moved.

$$\boldsymbol{x}_2 = \boldsymbol{M}_{2o}\boldsymbol{x}_{2o}\widetilde{\boldsymbol{M}}_{2o}, \tag{17.45}$$
$$\boldsymbol{x}_3 = \boldsymbol{M}_{2o}\boldsymbol{M}_{3o}\boldsymbol{x}_{3o}\widetilde{\boldsymbol{M}}_{3o}\widetilde{\boldsymbol{M}}_{2o}, \tag{17.46}$$
$$\boldsymbol{L}_3 = \boldsymbol{M}_{2o}\boldsymbol{L}_{3o}\widetilde{\boldsymbol{M}}_{2o}. \tag{17.47}$$

The point $\boldsymbol{x}_1 = \boldsymbol{x}_{1o}$ is not affected by the transformation, as is the case for the lines $\boldsymbol{L}_1 = \boldsymbol{L}_{1o}$ and $\boldsymbol{L}_2 = \boldsymbol{L}_{2o}$ (see Fig. 17.9). Since the rotation angle of the two axes $\boldsymbol{L}_2$ and $\boldsymbol{L}_3$ is related, we will use fractions of the angle q_1 to specify these rotation angles. The motors of each joint are computed using $\frac{2}{35}q_4$ to rotate around $\boldsymbol{L}_1$, $\frac{1}{125}q_1$ around $\boldsymbol{L}_2$, and $\frac{1}{375}q_1$ around $\boldsymbol{L}_3$. The angle coefficients were taken from the Barrett hand user manual:

$$\boldsymbol{M}_1 = \cos(q_4/35) + \sin(q_4/35)\boldsymbol{L}_1, \tag{17.48}$$
$$\boldsymbol{M}_2 = \cos(q_1/250) - \sin(q_1/250)\boldsymbol{L}_2, \tag{17.49}$$
$$\boldsymbol{M}_3 = \cos(q_1/750) - \sin(q_1/750)\boldsymbol{L}_3. \tag{17.50}$$

The position of each point is related to the angles q_1 and q_4 as follows:

$$\boldsymbol{x}'_1 = \boldsymbol{M}_1\boldsymbol{x}_1\widetilde{\boldsymbol{M}}_1, \tag{17.51}$$
$$\boldsymbol{x}'_2 = \boldsymbol{M}_1\boldsymbol{M}_2\boldsymbol{x}_2\widetilde{\boldsymbol{M}}_2\widetilde{\boldsymbol{M}}_1, \tag{17.52}$$
$$\boldsymbol{x}'_3 = \boldsymbol{M}_1\boldsymbol{M}_2\boldsymbol{M}_3\boldsymbol{x}_3\widetilde{\boldsymbol{M}}_3\widetilde{\boldsymbol{M}}_2\widetilde{\boldsymbol{M}}_1, \tag{17.53}$$
$$\boldsymbol{L}'_3 = \boldsymbol{M}_1\boldsymbol{M}_2\boldsymbol{L}_3\widetilde{\boldsymbol{M}}_2\widetilde{\boldsymbol{M}}_1, \tag{17.54}$$
$$\boldsymbol{L}'_2 = \boldsymbol{M}_1\boldsymbol{L}_2\widetilde{\boldsymbol{M}}_1. \tag{17.55}$$

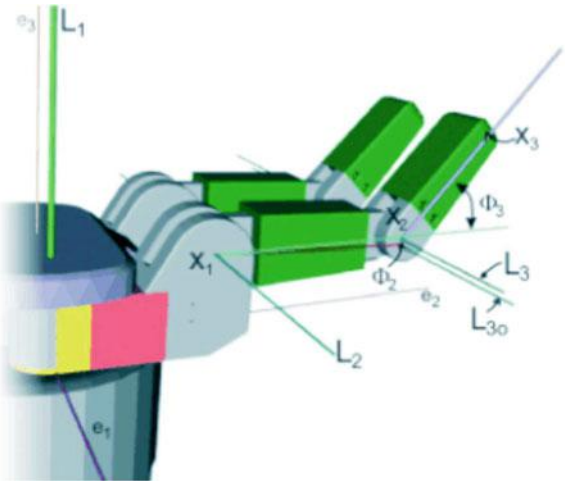


Fig. 17.9 Barrett hand at home position

Since we already know $\boldsymbol{x}'_3$, $\boldsymbol{L}'_1$, $\boldsymbol{L}'_2$, and $\boldsymbol{L}'_3$, we can calculate the speed of the end of the finger using Eq. (5.98) as follows:

$$\dot{\boldsymbol{X}}'_3 = \boldsymbol{X}'_3 \cdot \left(-\frac{2}{35}\boldsymbol{L}'_1\dot{q}_4 + \frac{1}{125}\boldsymbol{L}'_2\dot{q}_1 + \frac{1}{375}\boldsymbol{L}'_3\dot{q}_1 \right). \tag{17.56}$$

17.3 Pose Estimation

The pose estimation problem consists of determining the 3D position and orientation of the object. To estimate these parameters, the object in the image is segmented and the mathematical model of the object is projected onto the image to be compared with the segmented one. The difference between them is used to change the position and orientation of the object in order to reduce it. Figure 17.10 shows the closed-loop process (segmentation, comparison, and control) to estimate the pose of the object. Now each step of the process is explained.

17.3.1 Segmentation

Figure 17.11 shows the diagram of the evolution of the segmentation. Note that the object changes its color in the image. This is the result of the algorithm and it means that the center and boundaries of the object are known. The user chooses the object in the image by a double click in the image over a pixel of the object. This pixel is not necessarily in the center of the object. Suppose that the pixel is in the coordinates

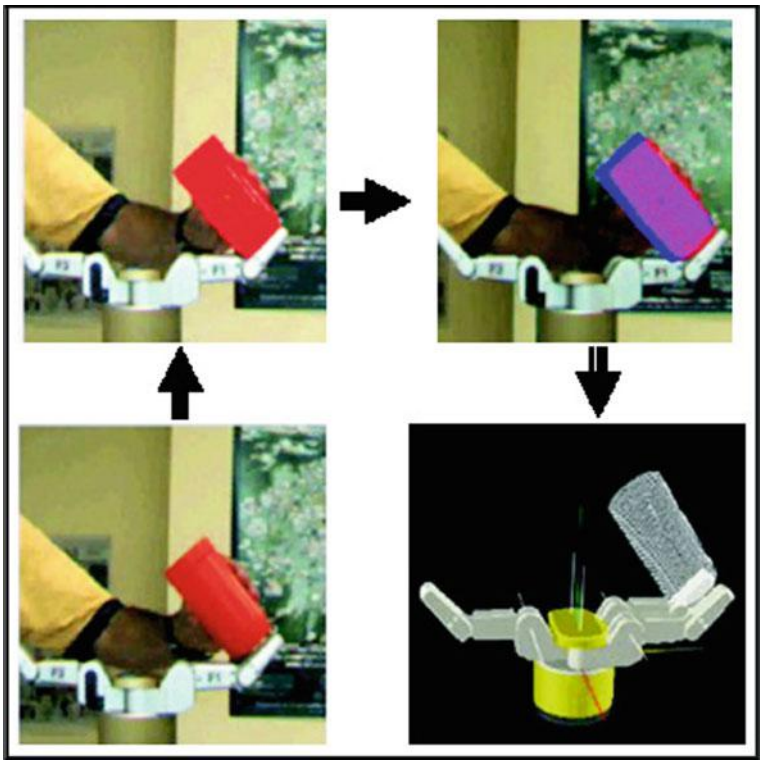


Fig. 17.10 Steps of the pose estimation algorithm

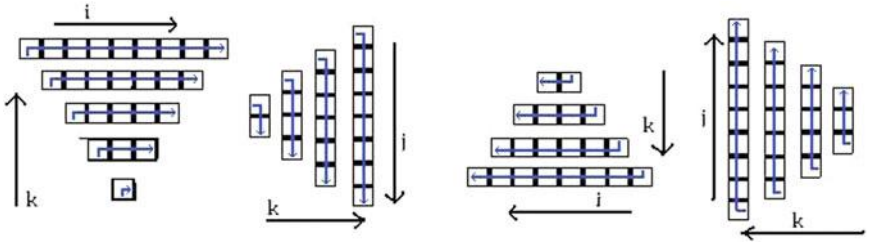


Fig. 17.11 Sequence of the segmentation algorithm

(x, y). Then the RGB color of this pixel is mapped to Hue–Saturation–Intensity (HSI) and memorized in $C_{x,y}$. The neighborhood of this pixel is explored, looking for similar pixels following the directions shown in Fig. 17.11. These correspond to the upper side:

$$
\begin{aligned}
&i \in \{x-k, \ldots, x+k\} \\
&\left(C_{i,y-k-1} \approx C_{x,y}\right) \cap \left((C_{i,y-k} = Red) \cup (C_{i-1,y-k-1} = Red)\right) \Rightarrow \\
&C_{i,y-k-1} = Red,
\end{aligned} \tag{17.57}
$$

the right side:

$$
\begin{aligned}
&j \in \{y-k-1, \ldots, y+k\} \\
&\left(C_{x+k+1,j} \approx C_{x,y}\right) \cap \left((C_{x+k,j} = Red) \cup (C_{x+k+1,j-1} = Red)\right) \Rightarrow \\
&C_{x+k+1,j} = Red,
\end{aligned} \tag{17.58}
$$

the lower side:

$$
\begin{aligned}
&i \in \{x+k+1, \ldots, x-k\} \\
&\left(C_{i,y+k+1} \approx C_{x,y}\right) \cap \left((C_{i,y+k} = Red) \cup (C_{i+1,y+k+1} = Red)\right) \Rightarrow \\
&C_{i,y-k-1} = Red,
\end{aligned} \tag{17.59}
$$

and the left side:

$$
\begin{aligned}
&j \in \{y+k+1, \ldots, y-k-1\} \\
&\left(C_{x-k-1,j} \approx C_{x,y}\right) \cap \left((C_{x-k,j} = Red) \cup (C_{x-k-1,j+1} = Red)\right) \Rightarrow \\
&C_{x-k-1,j} = Red.
\end{aligned} \tag{17.60}
$$

For example, Fig. 17.12 shows the segmentation of a rhombus. Of course, by codification, the number of computations can be reduced using flags to memorize the value of $C_{i-1,y-k-1}$ in the upper side; $C_{x+k+1,j-1}$ in the lower, etc. This algorithm labels the selected object in red.

17.3.2 Object Projection

Considering that by using cameras we can only see the surface of the observed objects, we consider them in this work as two-dimensional surfaces embedded in a 3D space and are described by the following function:

$$
H(s,t) = h_x(s,t)e_1 + h_y(s,t)e_2 + h_z(s,t)e_3, \tag{17.61}
$$

where s and t are real parameters in the range [0, 1]. Such a parametrization allows us to work with different objects like points, conics, quadrics, or even more complex real objects like cups, glasses, etc. (Table 17.1).

Computing the center of mass of the segmented object on each image (left and right) and retro-projecting the object lines onto left and right images, one can estimate the 3D position of the object. This is a useful approximation for the initialization of

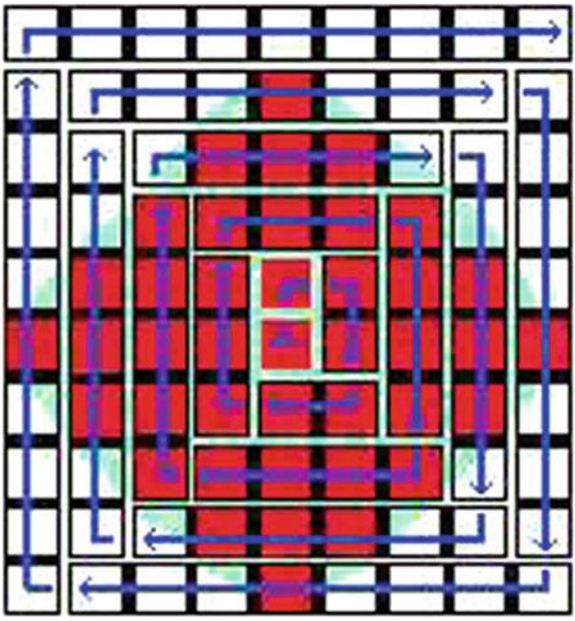


Fig. 17.12 Sequence of the segmentation algorithm

Table 17.1 Functions of some objects

Particle	$H = 3e_1 + 4e_2 + 5e_3$
Cylinder	$H = \cos(t)e_1 + \sin(t)e_2 + se_3$
Plane	$H = te_1 + se_2 + (3s + 4t + 2)e_3$

the algorithm. Next, we must consider that the object is in general position, which means that the object is rotated and translated with respect to the coordinate frame attached to the center of the left camera (see Fig. 17.13):

$$H'(s,t) = \boldsymbol{T}\boldsymbol{R}H(s,t)\tilde{\boldsymbol{R}}\tilde{\boldsymbol{T}}, \tag{17.62}$$

where

$$\boldsymbol{R} = \cos\left(\frac{\theta}{2}\right) - \sin\left(\frac{\theta}{2}\right)l = e^{-\frac{\theta}{2}l}, \tag{17.63}$$

$$\boldsymbol{T} = 1 + \frac{1}{2}ae_\infty = e^{-\frac{a}{2}e_\infty}, \tag{17.64}$$

where l is the rotation axis, θ is the rotation angle, and a is a translation vector. We need to estimate these transformations in order to know the object's pose. To estimate these parameters, we project the known mathematical model of the object ($H'(s,t)$) on the camera's image. This is possible because after calibration, we know the intrinsic parameters of the camera:

$$x = \frac{\pi_1 \cdot H'(s,t)}{\pi_3 \cdot H'(s,t)}, \tag{17.65}$$

$$y = \frac{\pi_2 \cdot H'(s,t)}{\pi_3 \cdot H'(s,t)}. \tag{17.66}$$

Note that the projective depth or translation (in the motor expression a) is initialized with the result of the triangulation with respect to the mass centers. The image of the mathematically projected model is compared with the image of the segmented object (see Fig. 17.13). If we find a match between them, then this means that the mathematical object is placed in the same position and orientation as the real object. Otherwise, we follow a descendant gradient-based algorithm to rotate and translate the mathematical model in order to reduce the error between them. This algorithm runs very fast.

Figure 17.14 shows the pose estimation result. In this case, we have a maximum error of 0.4° in the orientation estimation and 5 mm of maximum error in the estimated position of the object. The problem becomes more difficult to solve when the stereoscopic system is moving. Figure 17.15 shows how well the stereo system tracks the object. If we want to know the real object's position with respect to the world coordinate system, we must, of course, know the extrinsic camera's parameters. Figure 17.16 illustrates the object's position and orientation with respect to the robot's hand. In the upper row of this figure, we can see an augmented reality position sequence of the object. This shows that we can add the mathematical object to the real image. Furthermore, in the second row of the same image, we can see the virtual reality pose estimation result.

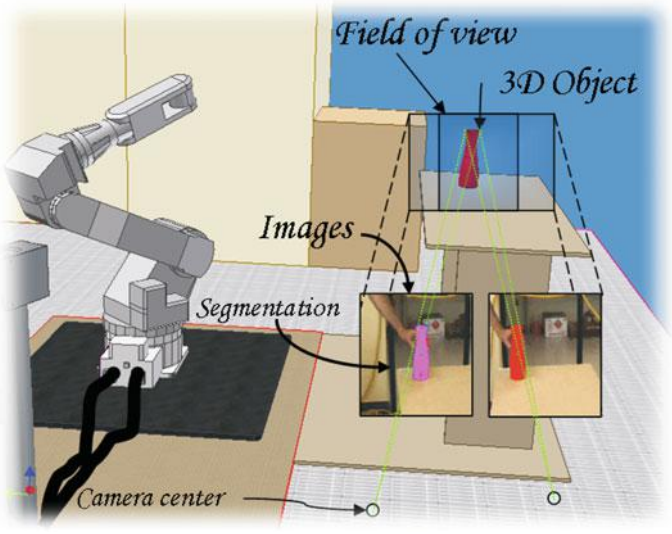


Fig. 17.13 Mathematical model of the object

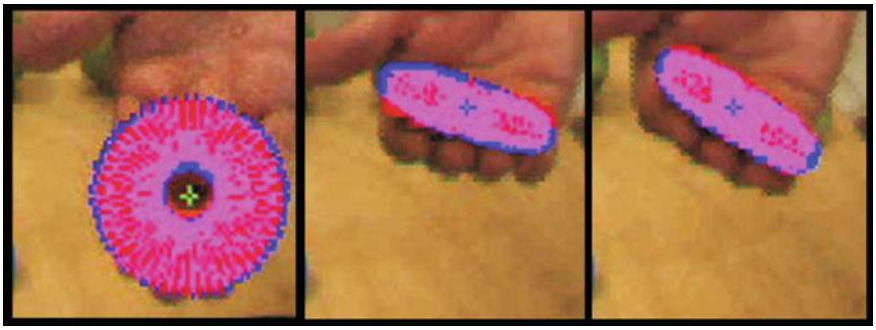


Fig. 17.14 Pose estimation of a disk with a fixed camera

Fig. 17.15 Pose estimation of a pot

17.4 Grasping Objects

There are many styles of grasping. However, we are taking into account only three principal styles. Also, note that for each style of grasping there are many possible solutions: for another approach, see [3].

17.4.1 First Style of Grasping

Since our objective is to grasp objects with the Barrett Hand, we must consider that it has only three fingers. So the problem consists of finding three points of grasping by which the system can be held in equilibrium. This means that the sum of the forces equals zero, as well as the sum of the moments.

We know the surface of the object, and so we can compute its normal vector in each point using

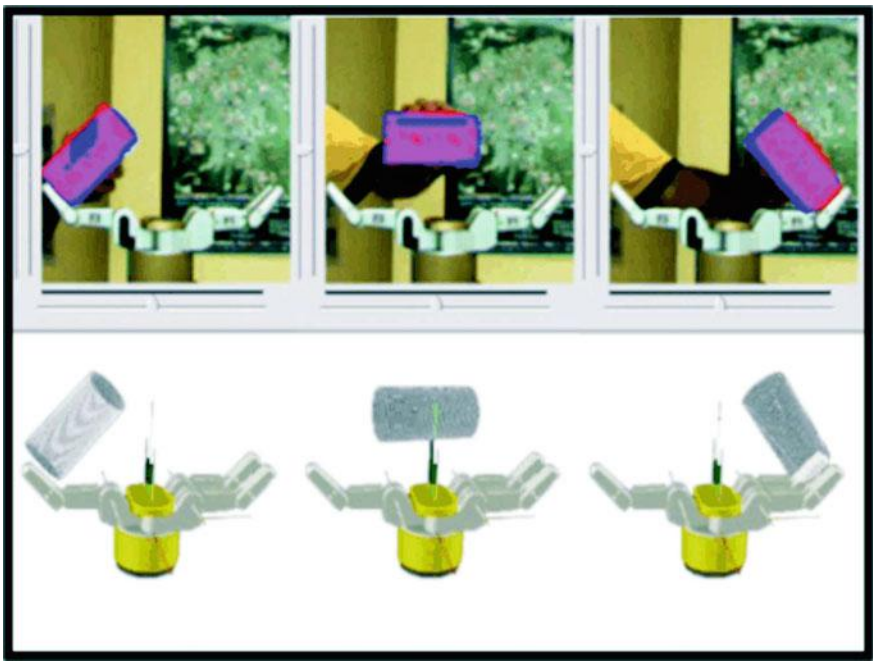


Fig. 17.16 Object presented in augmented and virtual reality

$$N(s,t) = \left(\frac{\partial \vec{H}(s,t)}{\partial s} \wedge \frac{\partial \vec{H}(s,t)}{\partial t} \right) I_e. \tag{17.67}$$

For surfaces with low friction, the value of F has to be closed to its projection over the normal ($F \approx F_n$). To maintain equilibrium, the sum of the forces must be zero, $\sum_{i=1}^{3} \|F_n\| N(s_i, t_i) = 0$ (Fig. 17.17).

This fact restricts the points over the surface in which the forces can be applied. This number of points is more reduced if we consider that the forces of equal magnitude over the object:

$$\sum_{i=1}^{3} N(s_i, t_i) = 0. \tag{17.68}$$

Additionally, in order to maintain the system's equilibrium, the sum of the moments must equal zero:

$$\sum_{i=1}^{3} H(s,t) \wedge N(s,t) = 0. \tag{17.69}$$

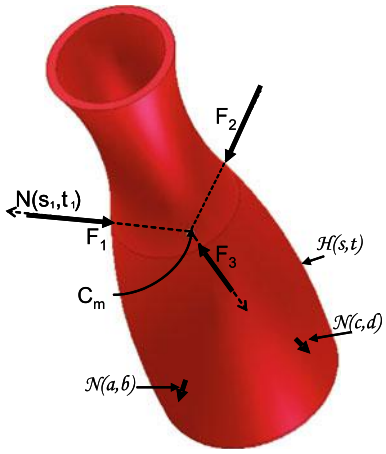


Fig. 17.17 Object with normal vectors

The points on the surface with the maximum and minimum distance to the mass center of the object fulfill $H(s,t) \wedge N(s,t) = 0$. The normal vector in such points crosses the center of mass (C_m) and does not produce any moment. Before determining the external and internal points, we must compute the center of mass as

$$C_m = \int_0^1 \int_0^1 \vec{H}\,(s,t)dsdt. \tag{17.70}$$

Once C_m is calculated, we can establish the next restriction:

$$(H(s,t) - C_m) \wedge N(s,t) = 0. \tag{17.71}$$

The values s and t satisfying (17.71) form a subspace of critical points $H(s,t)$ on the surface (being maxima, minima, or points of inflections).

The constraint imposing that the three forces must be equal is difficult to fulfill because it implies that the three points must be symmetric with respect to the center of mass. When such points are not present, we can relax the constraint to demand that only two forces are equal in order to fulfill the hand's kinematics equations. Then the normals $N(s_1,t_1)$ and $N(s_2,t_2)$ must be symmetric with respect to $N(s_3,t_3)$:

$$N(s_3,t_3)N(s_1,t_1)N(s_3,t_3)^{-1} = N(s_2,t_2). \tag{17.72}$$

17.4.2 Second Style of Grasping

In the first style of grasping, three points of contact were considered. In this section, we are taking into account a greater number of contact points. This fact generates a

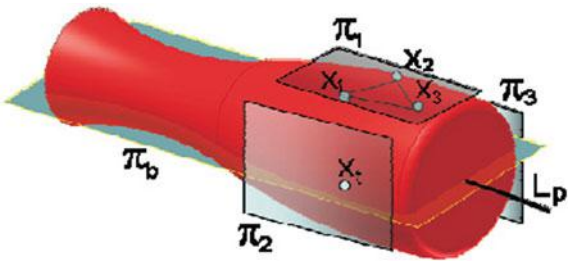


Fig. 17.18 Planes of the object

style of grasping that holds the objects more securely. To increment the number of contact points, the base of the hand is taken into account.

Since the object is described by the equation $H(s,t)$, it is possible to compute a plane π_b that cuts the object in the middle. This is possible using object points referred to the principal axis $\boldsymbol{L}_p$ and linear regression (see Fig. 17.18). One selects only the points from locations with their normal parallel to the plane π_b:

$$N(s,t) \wedge \pi_b \approx 0. \tag{17.73}$$

Now we choose three points of the object that are separated from another by 25 mm, to generate a plane on the object (see top plane in Fig. 17.18). In this style of grasping, the position of the hand relative to the object is trivial, because we just need to align the center of these points with the center of the hand's base. Also, the orientation corresponds to the normal of the plane, $\pi_1 = \boldsymbol{x}_1 \wedge \boldsymbol{x}_2 \wedge \boldsymbol{x}_3 \wedge e_\infty$.

17.4.3 Third Style of Grasping

In this style of grasping, the forces F_1, F_2, and F_3 do not intersect with the mass center. They are canceled by symmetry because the forces are parallel:

$$N(s_3,t_3)F_3 + N(s_1,t_1)F_1 + N(s_2,t_2)F_2 = 0. \tag{17.74}$$

Also, the magnitude forces F_1, F_2, and F_3 are in the plane π_b and are orthogonal to the principal axis $\boldsymbol{L}_p$ ($\pi_b = \boldsymbol{L}_p \cdot N(s,t)$), as you can see in Fig. 17.19.

A new restriction is then added to reduce the subspace of solutions:

$$F_3 = 2F_1 = 2F_2, \tag{17.75}$$

$$N(s_1,t_1) = N(s_2,t_2) = -N(s_3,t_3). \tag{17.76}$$

Finally, let us consider three points $\boldsymbol{x}_1$, $\boldsymbol{x}_2$, and $\boldsymbol{x}_3$ lying on the parallels depicted in Fig. 17.19. The directed distance between the parallels applied to the points $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ must be equal to 50 mm and that between $\boldsymbol{x}_1, \boldsymbol{x}_2$, to $\boldsymbol{x}_3$ must be equal to 25 mm.

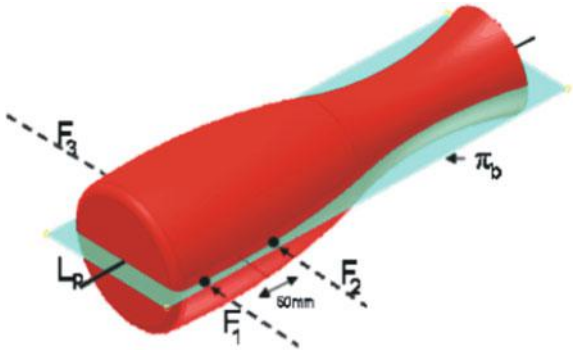


Fig. 17.19 Forces of grasping

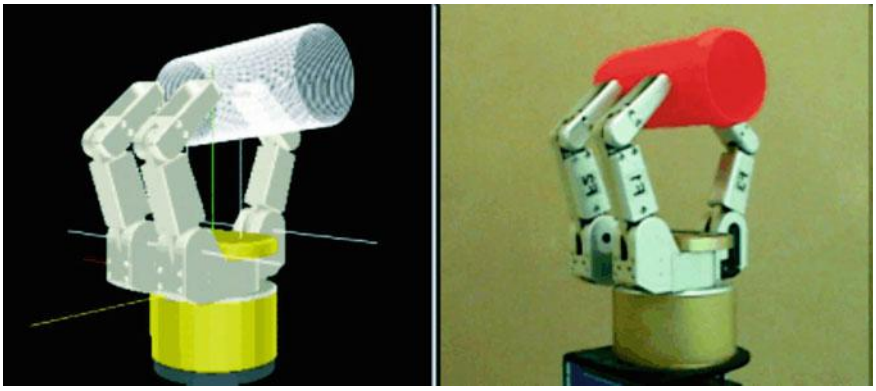


Fig. 17.20 Simulation and result of the grasping

Now we search exhaustively for the three points, changing s_i and t_i. Figure 17.20 shows the simulation and the result of this grasping algorithm. The position of the object relative to the hand must be computed using a coordinate frame in the object and the other frame in the hand.

17.5 Target Pose

Once the three grasping points ($P_1 = H(s_1, t_1)$, $P_2 = H(s_2, t_2)$, and $P_3 = H(s_3, t_3)$) are calculated, it is simple to determine the angles at the joints for each finger. To determine the angle of the spread ($q_4 = \beta$), we use

$$\cos\beta = \frac{(P_1 - C_m) \cdot (C_m - P_3)}{|P_1 - c_m|\,|C_m - P_3|}. \tag{17.77}$$

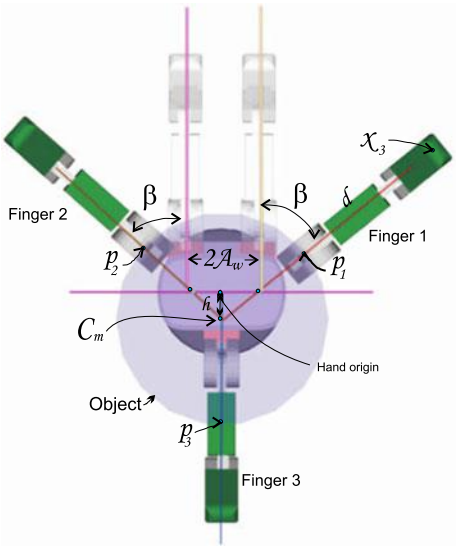


Fig. 17.21 Object's position relative to the hand

To calculate each of the finger angles, we determine its elongation as

$$\boldsymbol{x}_3' \cdot e_2 = |(P_1 - C_m)| - \frac{A_w}{\sin(\beta)} - A_1, \tag{17.78}$$

$$\boldsymbol{x}_3' \cdot e_2 = |(P_2 - C_m)| - \frac{A_w}{\sin(\beta)} - A_1, \tag{17.79}$$

$$\boldsymbol{x}_3' \cdot e_2 = |(P_3 - C_m)| + h - A_1, \tag{17.80}$$

where $\boldsymbol{x}_3' \cdot e_2$ determines the opening distance of the finger:

$$\boldsymbol{x}_3' \cdot e_2 = (M_2 M_3 \boldsymbol{x}_3 \widetilde{M}_3 \widetilde{M}_2) \cdot e_2, \tag{17.81}$$

$$\boldsymbol{x}_3' \cdot e_2 = A_1 + A_2 \cos\left(\tfrac{1}{125} q + I_2\right) + + A_3 \cos\left(\tfrac{4}{375} q + I_2 + I_3\right), \tag{17.82}$$

where e_2 is a unit vector pointing at the second finger joint (see Fig. 17.21). Solving for the angle q we have the opening angle for each finger. These angles are computed offline for each style of grasping of each object. They are the target in the velocity control of the hand.

17.5.1 Object Pose

We must find the transformation M, which allows us to put the hand in a way such that each finger end coincides with the corresponding contact point. For the sake of simplicity, the transformation M is divided into three transformations (M_1, M_2, M_3).

With the same purpose, we label the finger ends as $\boldsymbol{X}_1$, $\boldsymbol{X}_2$, and $\boldsymbol{X}_3$, and the contact points as P_1, P_2, and P_3.

The first transformation M_1 is the translation between the object and the hand, which is equal to the directed distance between the centers of the circles named $Z_h^* = \boldsymbol{X}_1 \wedge \boldsymbol{X}_2 \wedge \boldsymbol{X}_3$ and $Z_o^* = \boldsymbol{P}_1 \wedge \boldsymbol{P}_2 \wedge \boldsymbol{P}_3$, and it can be calculated as

$$\boldsymbol{M}_1 = e^{-\frac{1}{2}\left(\frac{Z_h^*}{Z_h^* \wedge e_\infty} \wedge \frac{Z_o^*}{Z_o^* \wedge e_\infty} \wedge e_\infty\right) I}. \tag{17.83}$$

The second transformation allows the alignment of the planes $\pi_h^* = Z_h^* \wedge e_\infty = \boldsymbol{X}_1 \wedge \boldsymbol{X}_2 \wedge \boldsymbol{X}_3 \wedge e_\infty$ and $\pi_o^* = Z_o^* \wedge e_\infty$, which are generated by the new points of the hand and the object. This transformation is calculated as $M_2 = e^{-\frac{1}{2}\pi_h \wedge \pi_o}$. The third transformation allows the points to overlap and can be calculated using the planes $\pi_1^* = Z_o \wedge \boldsymbol{X}_3 \wedge e_\infty$ and $\pi_2^* = Z_o \wedge \boldsymbol{P}_3 \wedge e_\infty$, which are generated by the circle's axis and any of the points $\boldsymbol{M}_3 = e^{-\frac{1}{2}\pi_1 \wedge \pi_2}$.

These transformations also define the pose of the object with respect to the hand. They are computed offline in order to know the target position and orientation of the object with respect to the hand and are used to design a control law for visually guided grasping.

17.6 Visually Guided Grasping

Once the target position and orientation of the object are known for each style of grasping and the hand's posture (angles of joints), it is possible to write a control law using this information and the equation of differential kinematics of the hand that allows one to grasp the object using visual guidance.

Basically, the control algorithm takes the pose of the object estimated as shown in Sect. 17.3 and compares it with each one of the target poses computed in Sect. 17.5 in order to choose the closest pose as the target. This way, the style of grasping is automatically chosen.

Once the style of grasping is chosen and the target pose is known, the error ϵ (between the estimated and the computed) is used to compute the desired angles in the joints of the hand:

$$\alpha_d = \alpha_t e^{-\epsilon^2} + (1 - e^{-\epsilon^2})\alpha_a, \tag{17.84}$$

where α_d is the desired angle of the finger, α_t is the target angle computed in Sect. 17.5, and α_a is the actual angle of the finger. Now the error between the desired position and the actual position is used to compute the new joint angle using the equation of differential kinematics of the Barrett hand given in Sect. 17.2.

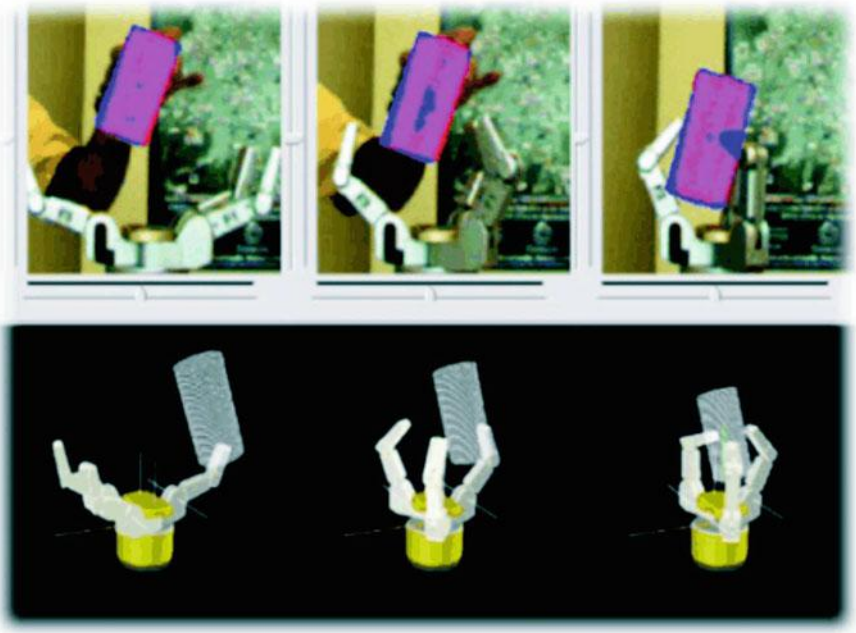


Fig. 17.22 Visually guided grasping

17.6.1 Results

Next, we show the results of the combination of the algorithms of pose estimation, visual control, and grasping to create a new algorithm for visually guided grasping. In Fig. 17.22, a sequence of images of the grasping is presented. When the bottle is carried in by the hand, the fingers are looking for possible points to grasp it. Now we can change the object or the pose of the object, and the algorithm then computes a new type of grasping. Figure 17.23 shows a sequence of images changing the object's pose.

17.7 Fuzzy Logic and Conformal Geometric Algebra for Grasping

A system in a closed loop is used to adjust the output variable of the process to control a reference input. To accomplish that, it is necessary to feed back the output variable of the process to a controller (see Fig. 17.24).

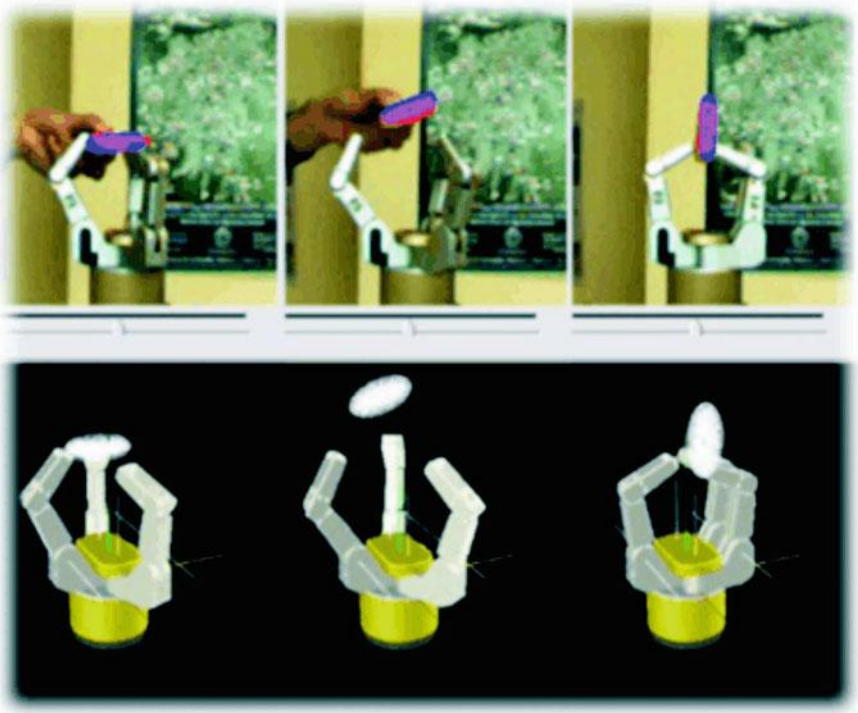


Fig. 17.23 Changing the object's pose

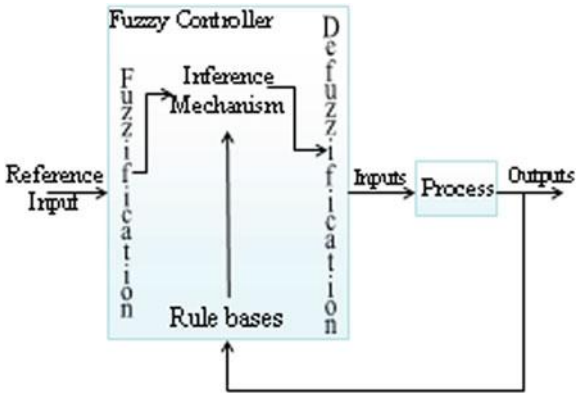


Fig. 17.24 A system with fuzzy controller

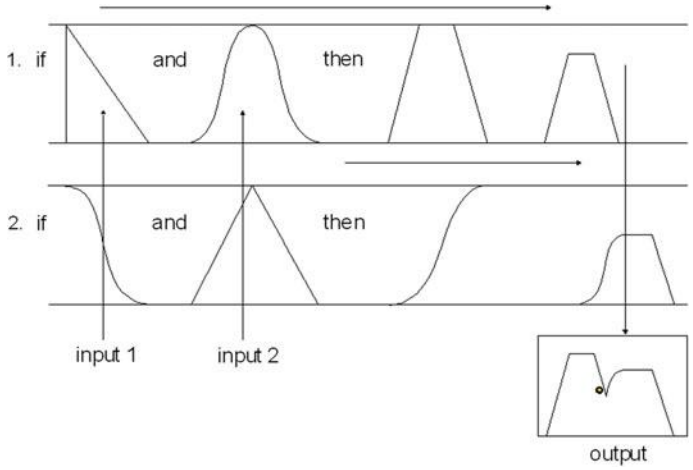


Fig. 17.25 Fuzzy inference diagram for a Mandami system

17.7.1 Mandami Fuzzy System

A Mandami fuzzy system [4] is characterized by the extensive use of geometric forms and heuristic methods to solve problems in real time. Such a system is comprised of the following elements: *fuzzification*: the input variables are fuzzified by the membership functions. These can be trapezoids, triangles, Gaussians, and others. *Rule bases* is a rule sets "if-then" where is stored the human knowledge base about the problem to be solved. For example, *if* object is **very big** *then* open **very much** the hand. *Inference mechanism* is responsible to process fuzzy rules of interest; for a reference input, it is possible that many rules can be fired. *Defuzzification*: this can be realized by many methods, including Centroid, Bisector, SoM, MoM, LoM, Med. Geom. Figure 17.25 shows the fuzzy inference diagram for a Mandami System (MS).

17.7.2 Direct Kinematics of the Barrett Hand

Figure 17.26 shows the Barrett Hand (BH) [5] in resting position. The BH has four motors, a motor for each finger (F1, F2, F3), each finger has an operation interval of [0°, 140°], and the other motor is for the simultaneous spread of F1 and F2 with an operation interval of [0°, 180°]. Figure 17.27 shows the calculation of the direct kinematics of F1 using its axis lines. The strategy consists of translating the parts of F1 to the origin, using $p_x(-A_1, 0, 0)$, $p_y(-A_2, 0, 0)$, $p_z(-A_3, D_3, 0)$. We can obtain the point $\boldsymbol{p}_k$, which corresponds to the fingertip for F1. Now we compute

Fig. 17.26 BH in resting position

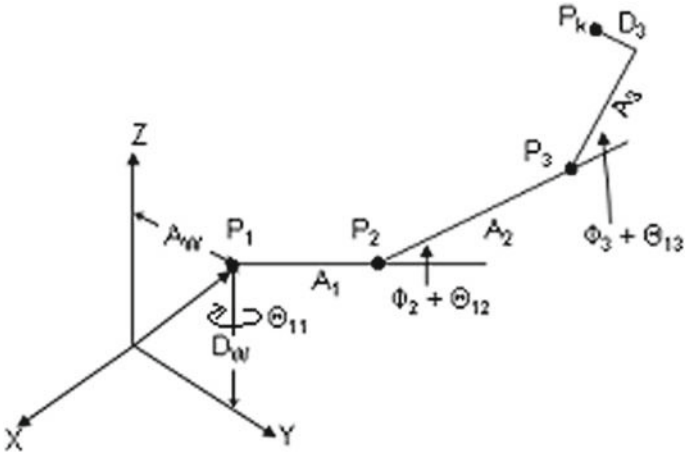


Fig. 17.27 Representation of F1 using lines

first the point related to the point $\mathbf{P}_x$: $\boldsymbol{p}_x = \mathbf{p}_x + \frac{\mathbf{p}_x^2}{2}e_\infty + e_0$, $\boldsymbol{t}_1 = A_W e_2 + D_W e_3$, $\boldsymbol{T}_1 = e^{\frac{1}{2}e_\infty \boldsymbol{t}_1}$, and $\boldsymbol{R}_1 = e^{\frac{1}{2}\theta_{11}e_1e_2}$. The equation of point $\boldsymbol{p}_2$ reads

$$\boldsymbol{p}_2 = \boldsymbol{T}_1\boldsymbol{R}_1\boldsymbol{p}_x\widetilde{\boldsymbol{R}}_1\widetilde{\boldsymbol{T}}_1.$$

The other fingers have an identical geometry. $\boldsymbol{p}_y = \mathbf{p}_y + \frac{\mathbf{p}_y^2}{2}e_\infty + e_0, \boldsymbol{t}_2 = \boldsymbol{p}_2, \boldsymbol{T}_2 = e^{\frac{1}{2}e_\infty \boldsymbol{t}_2}$, $\boldsymbol{R}_2 = e^{\frac{1}{2}(\theta_{12}+\Phi_2)e_2e_3}$. The equation of point $\boldsymbol{p}_3$ then reads

$$\boldsymbol{p}_3 = \boldsymbol{T}_2\boldsymbol{R}_1\boldsymbol{R}_2\boldsymbol{p}_y\widetilde{\boldsymbol{R}}_2\widetilde{\boldsymbol{R}}_1\widetilde{\boldsymbol{T}}_2,$$

$\boldsymbol{p}_z = \mathbf{p}_z + \frac{\mathbf{p}_z^2}{2}e_\infty + e_0, \boldsymbol{t}_3 = \boldsymbol{p}_3, \boldsymbol{T}_3 = e^{\frac{1}{2}e_\infty \boldsymbol{t}_3}$, $\boldsymbol{R}_3 = e^{\frac{1}{2}(\theta_{13}+\Phi_3)e_2e_3}$, where $\theta_{13} = \frac{1}{3}\theta_{12}$. Finally for the point $\boldsymbol{p}_k$, we write

$$\boldsymbol{p}_k = \boldsymbol{T}_3\boldsymbol{R}_1\boldsymbol{R}_2\boldsymbol{R}_3\boldsymbol{P}_z\widetilde{\boldsymbol{R}}_3\widetilde{\boldsymbol{R}}_2\widetilde{\boldsymbol{R}}_1\widetilde{\boldsymbol{T}}_3.$$

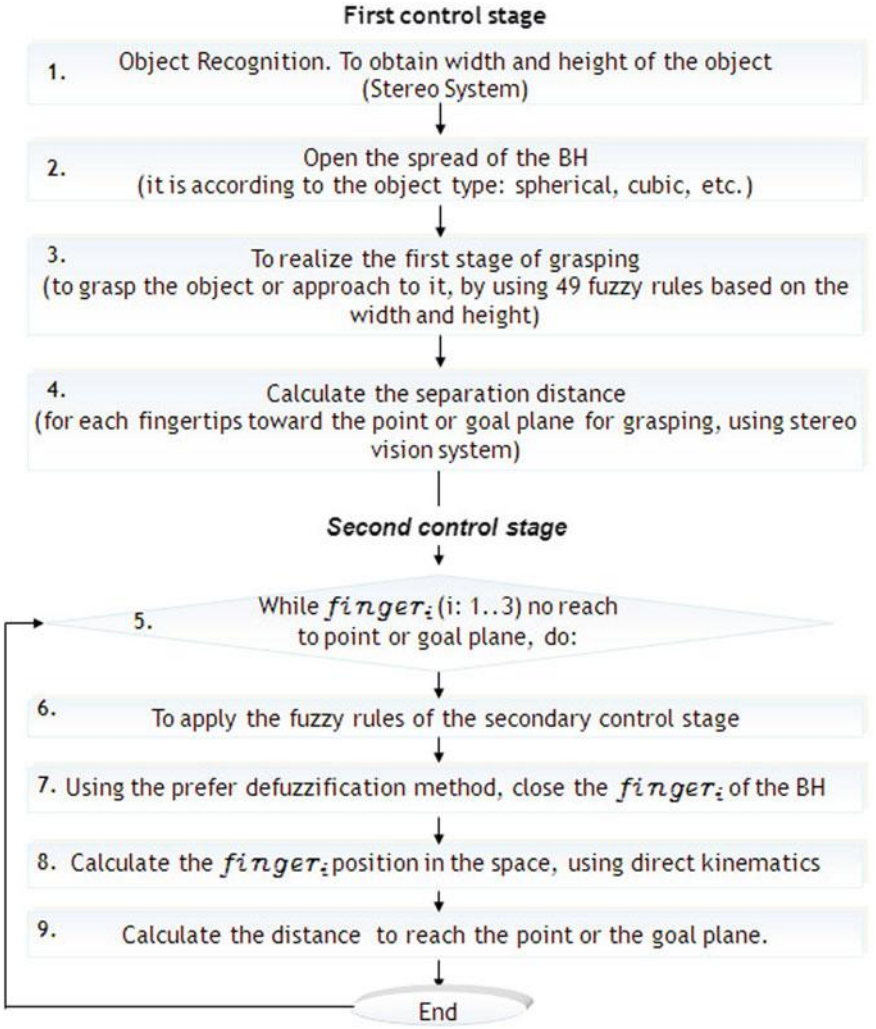

First control stage
1. Object Recognition. To obtain width and height of the object (Stereo System)
2. Open the spread of the BH (it is according to the object type: spherical, cubic, etc.)
3. To realize the first stage of grasping (to grasp the object or approach to it, by using 49 fuzzy rules based on the width and height)
4. Calculate the separation distance (for each fingertips toward the point or goal plane for grasping, using stereo vision system)
Second control stage
5. While $finger_i$ (i: 1..3) no reach to point or goal plane, do:
6. To apply the fuzzy rules of the secondary control stage
7. Using the prefer defuzzification method, close the $finger_i$ of the BH
8. Calculate the $finger_i$ position in the space, using direct kinematics
9. Calculate the distance to reach the point or the goal plane.
End

Fig. 17.28 Algorithm for grasping objects

17.7.3 Fussy Grasping of Objects

Figure 17.28 shows the algorithm used for grasping objects using FL and CGA. We numbered those stages in which we used FL and/or CGA. These are as follows: (1) CGA, (2) FL, (3) FL, (4) CGA: direct kinematics, (5) CGA: direct kinematics, (6) FL, (7) FL, (8) direct kinematics, (9) CGA. The first control stage consists of grasping the object or to come closer to it; it is based on the *Height and Width* of the object. To obtain the *Output Angle* of the fingers, we use 49 fuzzy rules shown in Table 17.2.

Table 17.2 Fuzzy rules for the first control stage

Fuzzy rules	VT	T	VS	S	M	L	VL
VT	VL	L	M	M	L	VS	VT
T	VL	L	M	M	L	VS	VT
VS	VL	L	L	L	M	VS	VT
S	VL	L	L	M	M	VS	VT
M	VL	L	L	M	M	S	T
L	VL	L	L	M	M	S	T
VL	VL	L	L	M	L	S	T

The fuzzy variables, Height and Width, of the object are shown in the rows and columns, respectively. The abbreviations used are very tiny (VT), tiny (T), very small (VS), small (S), medium (M), large (L), very large (VL). An example of these rules is *if* the height is VT and the width is VT *then* Output Angle for all finger is VL. The second control stage has five fuzzy rules, based on the *Separation distance* of the fingertips of each finger to its plane or point goal. These rules are *if* Separation is VL or L *then* the Output Angle is M, *if* Separation is M *then* the Output Angle is S, *if* Separation is VS or S *then* the Output Angle is VS, *if* Separation is T *then* the Output Angle is T, *if* Separation is VT *then* the Output Angle is VT.

The sphere was calculated with only four points on it. The position in 3D and the size of the sphere were calculated using CGA; for the cube, four points were taken to calculate the width, height, and depth. The 3D points are obtained using a stereo vision system (see Fig. 17.29c). Figure 17.29a–d shows the execution of the algorithm shown in Fig. 17.28 in virtual reality and in real time.

17.8 Conclusion

This chapter applies a powerful geometric language for the development of theory, concepts, and computer algorithms in the domain of robot vision. As opposed to current approaches, all the computations are done in the conformal geometric framework, and there is no need to change the mathematical framework. In order to show the power of this flexible mathematical language, we developed real-time algorithms for a real scenario of perception, approach, and action that handles a number of real robot manipulation tasks. This chapter shows a feasible grasping strategy based on the mathematical model of the object and that of the manipulator. We compute the 3D poses of an object and the robot hand necessary to close the loop among perception, approaching, and action. A control law with visual feedback has been developed to grasp objects using the mechanical and visual Jacobian matrix in terms of the line axis of the Barrett hand and the principal optical planes of the camera. We show also an interesting application of fuzzy logic and conformal geometric algebra for

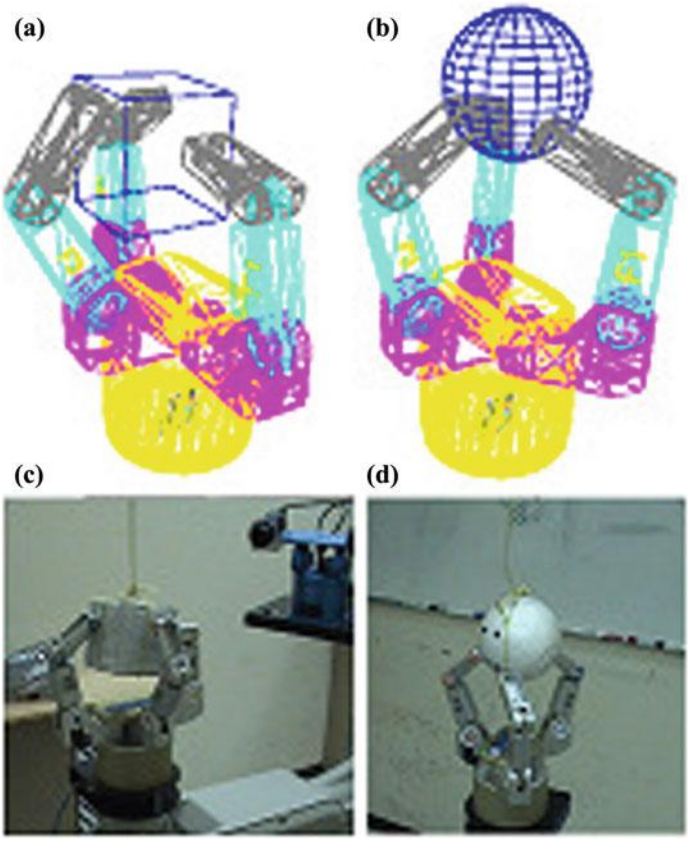


Fig. 17.29 Example of execution of the algorithm (**a**–**d**). Stereo vision system (**c**)

grasping using the Barrett hand. For the fuzzy control of the hand, we implemented a Mandami fuzzy system.

Our applications in a real scenario prove the efficiency of algorithms developed using conformal geometric algebra. By simplifying the geometric algorithms at a symbolic level and accelerating their execution using cost-effective hardware, one should reduce the execution time in great proportions.

References

1. Hartley, R. (1993). Chirality invariants. In *DARPA Image Understanding Workshop* (pp. 745–753).
2. Siciliano, B. (1990). *Robot kinematics* (Vol. 1, pp. 1–50). Università degli di napoli, Italy. CRC Press Handbook.

3. Borst, C., Fischer, M., & Hirzinger, G. (1999). A fast and robust Grasp planner for arbitrary 3D objects. In *International Conference on Robotics and Automation ICRA99* (pp. 1890–1896).
4. Passino, K. M. (1998). *Fuzzy control*. Reading, MA: Addison-Wesley.
5. Barrett-Technology. http://www.barrett.com/robot/index.htm.

Chapter 18
3D Maps, Navigation, and Relocalization

This chapter presents applications of body–eye calibration algorithms using motors of the conformal geometric algebra. A scan-matching algorithm, based on such algorithm, aligns the scans by representing the scan points as lines. We show then a path-following procedure that also uses the conformal geometric algebra techniques to estimate the geometric error for a control law. The chapter extends the applications of body-sensor calibration for the case of stereo vision and laser scanner. Such sensors are used for building 3D maps and tackling the relocalization problem. For the relocalization, we resort to an approach based in the Hough transform, where the desired position is searched in the line Hough space.

18.1 Map Building

One of the most important tasks in mobile robotics is the tracking of the robot pose. To solve this problem, we use a laser sensor mounted on the robot and a scan-matching algorithm to estimate the robot pose using laser scans.

A laser scan is a set of points corresponding to the intersection between the laser light and the surrounding objects. Each measurement is represented in a polar coordinate system whose origin is the laser origin; the magnitude of the measurement represents the distance between the object and the sensor (Fig. 18.1).

18.1.1 Matching Laser Readings

Given a reference scan S_{ref}, a new scan S_{new}, and a rough estimation (θ_0, x_0, y_0) of the relative displacement of the sensor between the scans, the objective of the matching problem is to estimate the real displacement (θ, x, y) between them (Fig. 18.2). As we saw in Eq. (6.30), the inner product of two points $\boldsymbol{X}$ and $\boldsymbol{Y}$ represents the Euclidean distance between the points $\mathbf{x}$ and $\mathbf{y}$, that is,

© Springer Nature Switzerland AG 2020

E. Bayro-Corrochano, *Geometric Algebra Applications Vol. II*,
https://doi.org/10.1007/978-3-030-34978-3_18

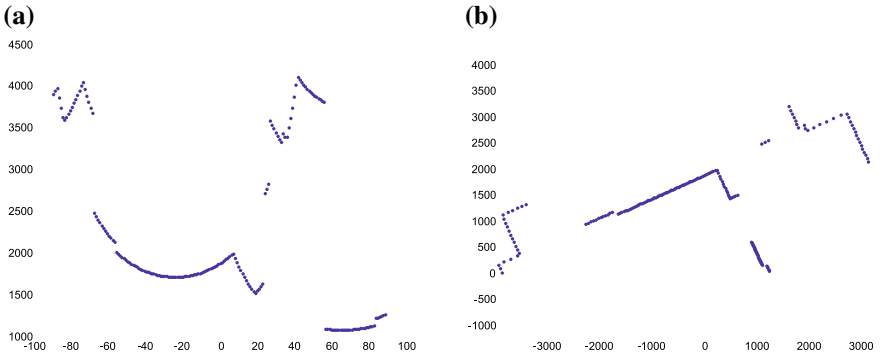


Fig. 18.1 **a** Laser readings in polar coordinates. **b** Laser readings in rectangular coordinates

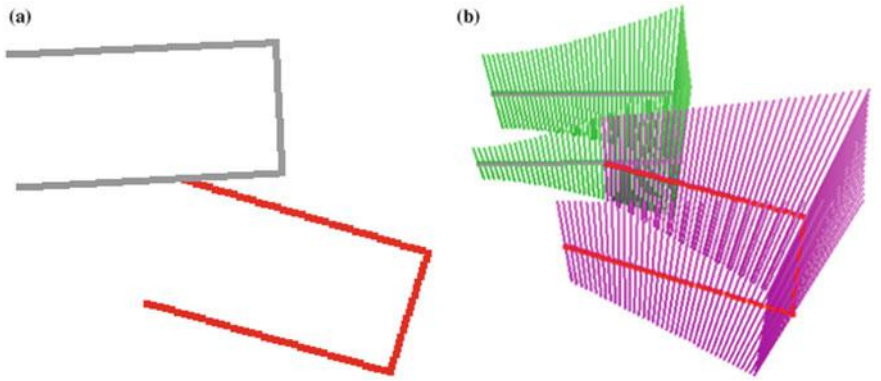


Fig. 18.2 **a** A pair of laser readings that must be aligned. **b** The lines passing through the laser reading points; these lines are orthogonal to the plane where the points lie. These lines can be interpreted as the motor axes of body–eye calibration problem, and thus we can use the same algorithm to solve the problem

$$\|\mathbf{x} - \mathbf{y}\| = \sqrt{-2(\boldsymbol{X} \cdot \boldsymbol{Y})}. \tag{18.1}$$

For speed reasons, we use directly the result of the product $\boldsymbol{X} \cdot \boldsymbol{Y}$. However, we should take into account the negative sign in (6.30), and thus instead of looking for a minimum, we look for a maximum to find the closest point to another point. The closest point can be defined as

$$closest(\boldsymbol{X}, \boldsymbol{Y}_i) = \max \boldsymbol{X} \cdot \boldsymbol{Y}_i \quad \text{for} \quad i \in (1, \ldots, N). \tag{18.2}$$

Once the correspondence between the points is known, we proceed to compute the rigid transformation between them. To solve this problem, we utilize the motor estimation algorithm of Sect. 16.2.2 used in the body-sensor calibration problem. We only need to express the points as lines orthogonal to the plane where they lie,

as if they were the axes of the motors in the body-sensor calibration problem. These lines can be found easily with

$$\boldsymbol{L}_i^* = \boldsymbol{X}_i \wedge e_3 \wedge e_\infty \quad \text{for} \quad i \in (1, \ldots, N). \tag{18.3}$$

In Fig. 18.2, we show the scans to be matched and the lines used to do it.

The algorithm to match two scans $\boldsymbol{X}_i$ and $\boldsymbol{Y}_j$ is shown below and can be initialized with odometry. Figure 18.3 shows some iterations for the alignment of the scans. Note that the algorithm can be initialized with odometry.

SCAN and MATCH Algorithm

1. For each point $\boldsymbol{X}_i$, estimate the closest point to the points $\boldsymbol{Y}_j$ using (18.2).
2. For each pair of correspondence points $(\boldsymbol{X}_i, \boldsymbol{X}_j)$, compute the lines $\boldsymbol{L}_i^*$ and $\boldsymbol{L}_j^*$ using (18.3).
3. From each pair of lines extract their moments $m_1 = \alpha_1 e_1 + \beta_1 e_2$ and $m_2 = \alpha_2 e_1 + \beta_2 e_2$.
4. With the elements α_1, β_1 y α_2, β_2, we add two files to the matrix A of Eq. (16.52).
5. Compute the SVD of A, $A = USV^T$ and choose the solution vector with the smallest singular value, say $v = v = \alpha_1 e_1 + \alpha_2 e_2 + \alpha_3 e_3 + \alpha_4 e_4$.
6. With the vector v compute the following motor: $\boldsymbol{M}_0 = \alpha_1 - \alpha_2 e_{12} + e_\infty(\alpha_3 e_1 + \alpha_4 e_2)$.
7. Normalize the motor $\boldsymbol{M}_o$ to get the solution motor, that is, $\boldsymbol{M} = \frac{\boldsymbol{M}_0}{\sqrt{\boldsymbol{M}_0 \widetilde{\boldsymbol{M}}_0}}$.
8. Update the points $\boldsymbol{Y}_j$ with the new motor, $\boldsymbol{Y}_j = \boldsymbol{M}\boldsymbol{Y}_j\widetilde{\boldsymbol{M}}$ (**for** $j \in 1 \ldots \boldsymbol{M}$).
9. If (error $>$ threshold) and (n $<$ max_iterations) go to step 1.

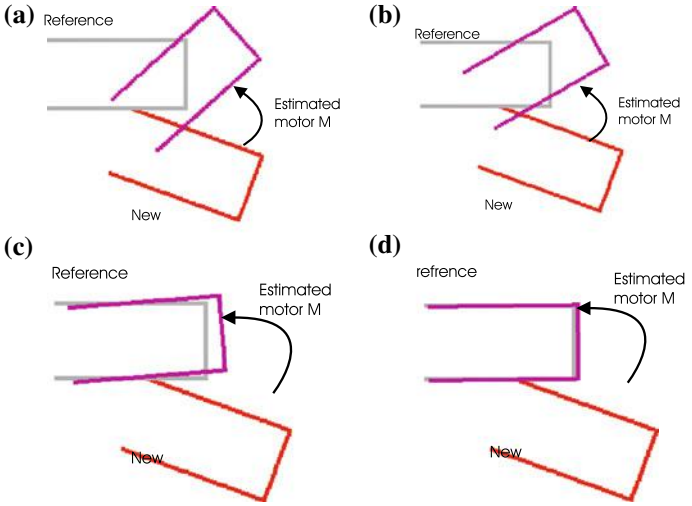


Fig. 18.3 Alignment of two laser scan references and new using the scan-matching algorithm. **a** After 5 iterations. **b** After 10 iterations. **c** After 15 iterations. **d** After 20 iterations

18.1.2 Map Building

Once we are able to match a pair of scans, we can build a global map. The global map is created using the motors computed from the scan-matching algorithm. Let $\boldsymbol{M}_0, \boldsymbol{M}_1, \ldots, \boldsymbol{M}_n$ be the motors obtained from the scan match that align the pair of scans $0, 1, \ldots, n$, respectively, and we define the global motor as

$$\boldsymbol{M} = \boldsymbol{M}_n \boldsymbol{M}_{n-1} \ldots \boldsymbol{M}_0. \tag{18.4}$$

This motor is applied to the n-scan in order to add it to the global map. In Fig. 18.4, we show how the matched scans are added to the global map.

18.1.3 Line Map

Once we have a 2D map, we proceed to build a line map, which is a map represented with a set of line segments. Each line segment represents a wall on the 3D world, that is, each line segment is the projection of the wall onto the floor. There are three main problems in line extraction in the map. They are as follows: How many lines are there? Which points belong to which line? Given the points that belong to a line,

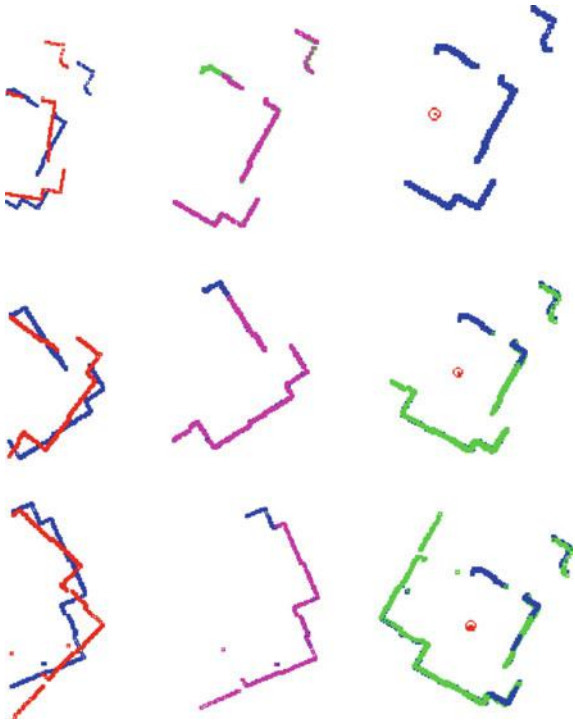


Fig. 18.4 Alignment of consecutive laser scans in order to build a global map

how to estimate the line model parameters? For the third problem, we use a common fitting algorithm, called *total least squares*, since it has been used extensively in the literature [1].

To solve the first two problems, we use the RANSAC (Random Sample Consensus) algorithm [2]. The RANSAC is an algorithm for robust fitting of models in the presence of data outliers, see the algorithm below. Figure 18.5 shows the resulting line map; observe that now the map is represented with line segments not with points.

RANSAC Algorithm

1. Initial: A set of N points.
2. Repeat.
3. Choose a sample of two points uniformly at random.
4. Compute the distances of other points to the line.
5. Construct the inlier set.
6. If there are enough inliers, recompute the line parameters, store the line, remove the inliers from the set.
7. until Max_iterations reached or too few points left.

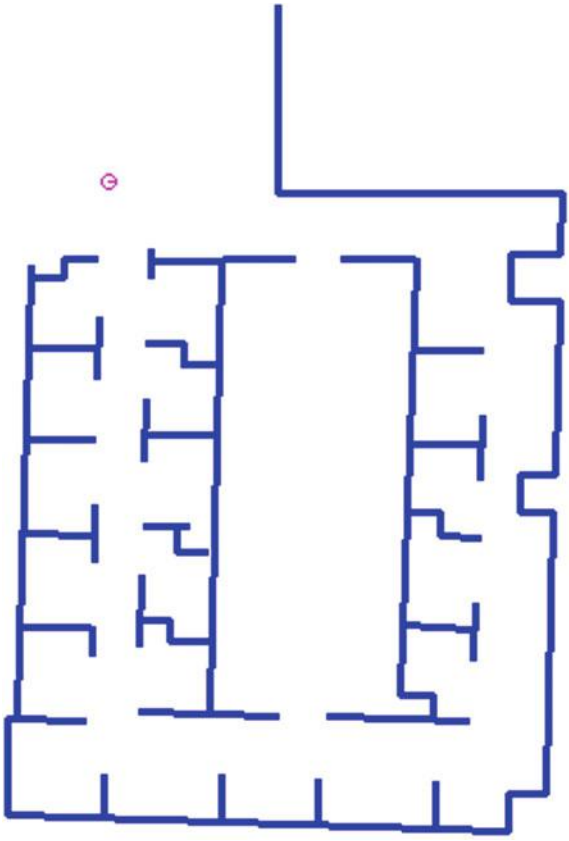


Fig. 18.5 Linear map built with the matched laser scans

18.1.4 3D Map Building

The line model can be used as a floor plane for the 3D map, where each line segment represents a wall (Fig. 18.6). Each line segment ($\boldsymbol{L}_i$) describes a plane, that is,

$$\Pi_i = \boldsymbol{L}_i \wedge e_3. \tag{18.5}$$

Once we have the walls of the 3D map, we proceed to add the texture. The texture is obtained from catadioptric images taken at the same time as the scans. Since the laser and the catadioptric camera are calibrated, we can project sections of the catadioptric image to the planes to add the texture. Since the mirror causes a distortion to the image, we must apply a processing to generate the texture correctly. The process consists of the generation of a perspective image from the catadioptric image, using the inverse point projection [3, Sect. 9.7.3]. Figure 18.7 shows a catadioptric image and a perspective image generated from it.

Once the perspective image is generated, it is projected to its corresponding plane. Let S be a sphere representing an omnidirectional camera and let $\boldsymbol{X}_s$ be a point on the sphere computed with the inverse point projection. To find the projection of the point onto the plane, we first compute the line that passes through the center of the sphere and the point $\boldsymbol{X}_s$ with

$$\boldsymbol{L}^* = S \wedge \boldsymbol{X}_s \wedge e_\infty. \tag{18.6}$$

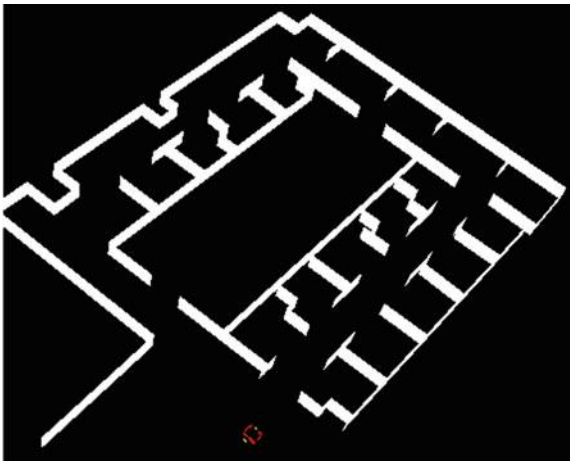


Fig. 18.6 3D map built with the linear map

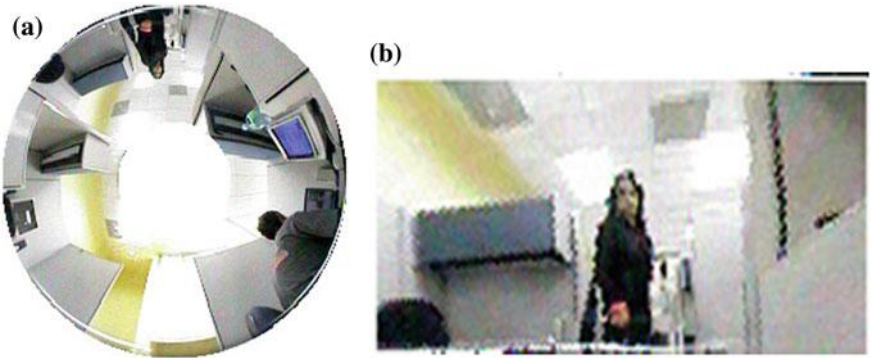


Fig. 18.7 **a** Omnidirectional image. **b** Perspective image generated from the omnidirectional image

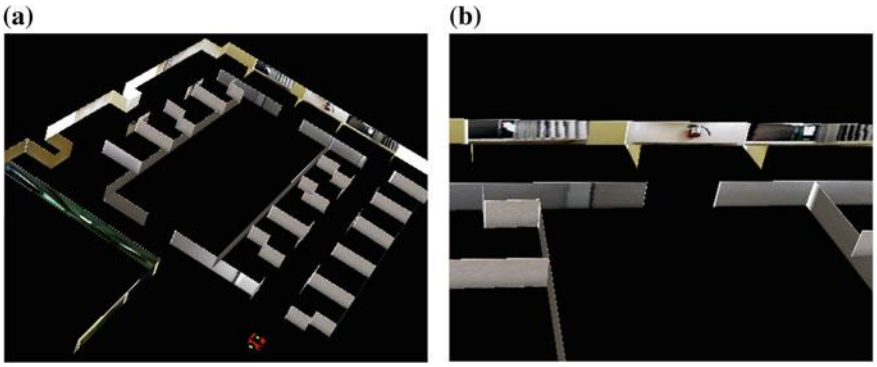


Fig. 18.8 **a** 3D map with texture, the texture was generated using the omnidirectional vision system, **b** a zoom of the 3D map

Then, compute the projection of the point $\boldsymbol{X}_s$ onto the plane Π as

$$\boldsymbol{X}_p = \Pi \cdot \boldsymbol{L}. \tag{18.7}$$

The computations are applied to all the points that will be added to the texture. A 3D map with texture is shown in Fig. 18.8a.

18.2 Navigation

The mobile robot must know its position and heading to implement a navigation task. Global localization is used to find the mobile robot's initial position. Local localization is used to track the mobile robot's position over time when the initial position of the mobile robot is known.

18.2.1 Localization

To solve the local localization problem, the robot estimates its ego-motion using pairs of readings and the scan-matching algorithm (see Sect. 18.1.1). Each scan matching gives us a motor that aligns two scans, and therefore for n matchings we have n motors, i.e., $\boldsymbol{M}_1, \boldsymbol{M}_2, \ldots, \boldsymbol{M}_n$. The actual pose of the laser sensor is given by the concatenation of n motors and the initial motor $\boldsymbol{M}_0$ obtained in the global localization. Thus, the motor representing the actual pose of the laser is defined as

$$\boldsymbol{M}_L = \boldsymbol{M}_n \boldsymbol{M}_{n-1} \ldots \boldsymbol{M}_0. \tag{18.8}$$

The actual pose of the robot can be found using the calibration motor $\boldsymbol{M}_x$ between the laser and the robot as

$$\boldsymbol{M}_r = \boldsymbol{M}_x \boldsymbol{M}_L. \tag{18.9}$$

The center of the robot can be found with

$$\boldsymbol{X}_o = \boldsymbol{M} e_0 \widetilde{\boldsymbol{M}}, \tag{18.10}$$

and the line representing its heading is

$$\boldsymbol{L}_h = \boldsymbol{M} e_1 E \widetilde{\boldsymbol{M}}. \tag{18.11}$$

With this information the robot can be localized on the map, and Fig. 18.9 shows the localization of the robot using only odometry and using the scan-matching algorithm.

18.2.2 Adding Objects to the 3D Map

Sometimes when the robot navigates, it can be necessary to add objects from the real world to the built map. The representation of the objects can be simple or detailed. In the simple representation, we use a sphere to indicate the position of an object on the map. Also, we store images of the object in order to know what represents the sphere. The figure shows the simulation of the built map, the pose of the robot, and the position of the object.

18.2.3 Path Following

The path-following task runs as follows: given a map trajectory the robot should follow it. To achieve the task, the robot must control its heading. The path is defined as a set of points on the map. The points are sampled close to each other and interpolated to smoothly follow the curve.

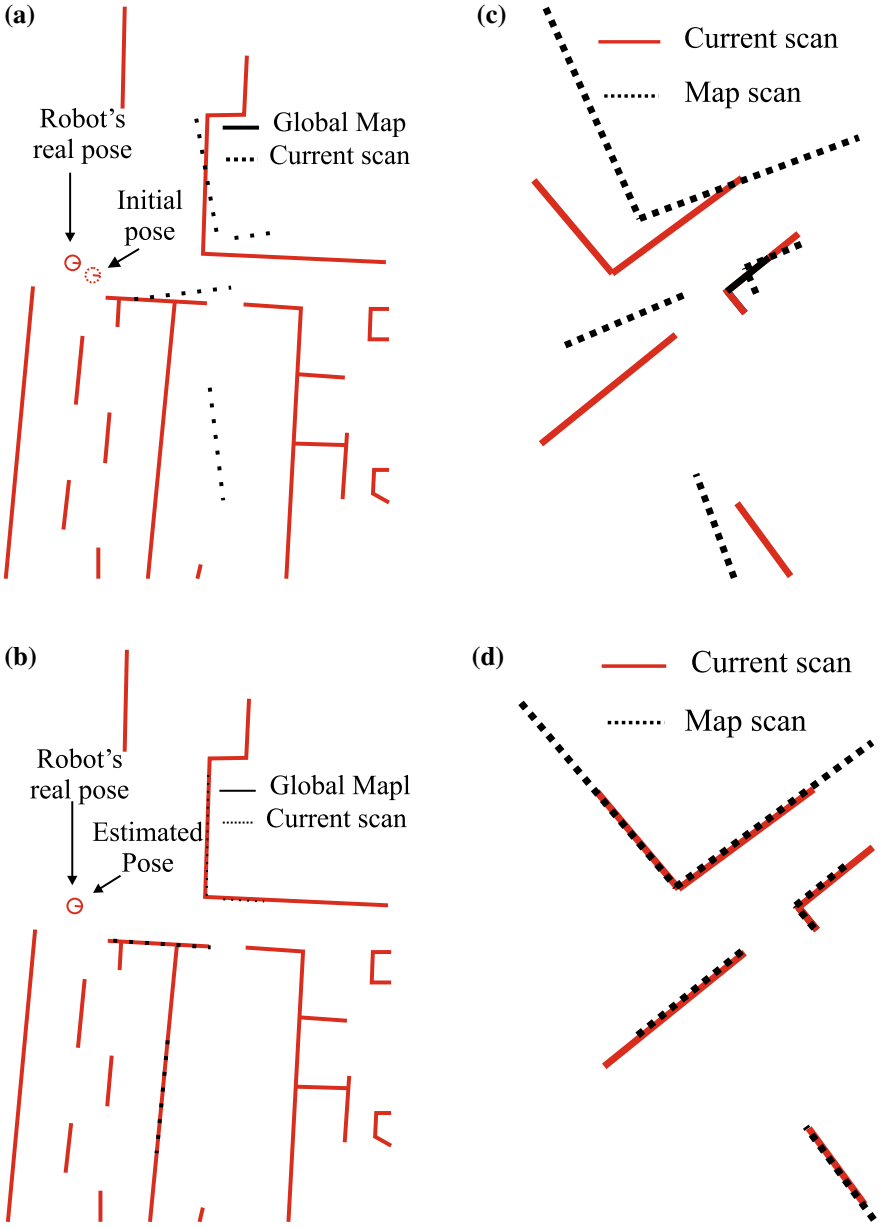


Fig. 18.9 Parts **a** and **b**, respectively, show the global map before and after the scan matching with respect to it. Parts **c** and **d** show the scan matching of the current scan with respect to a scan from the global map

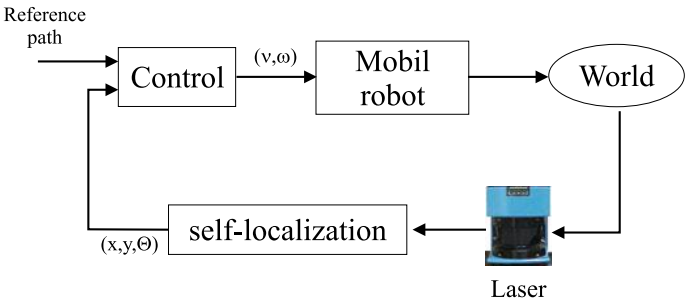


Fig. 18.10 Flowchart of the control system for the path-following task

The control system designed for the path-following task is based on a feedback control system encompassing a self-localization module and a control module (Fig. 18.10). The control module computes the commanding signals for the robot, namely, linear and angular velocities, based on the error distance obtained by comparing the current pose of the robot, estimated by the self-localization module, with the desired pose.

The kinematic model of our robot is described by

$$\begin{cases} \dot{x} = \nu \cos(\theta), \\ \dot{y} = \nu \sin(\theta), \\ \dot{\theta} = \quad \omega, \end{cases} \tag{18.12}$$

where the two velocity control inputs are the linear velocity v and the angular velocity ω.

The mobile robot and the path to be followed, denoted by $\mathbf{P}$, are represented in Fig. 18.11, where $\boldsymbol{X}_r$ is the orthogonal projection of the robot point $\boldsymbol{X}$ on $\mathbf{P}$. The line $\boldsymbol{L}_h$ represents the heading of the robot and the line $\boldsymbol{L}_r$ represents the line tangent to the path at $\boldsymbol{X}_r$. The signed distance between the point $\boldsymbol{X}_r$ and $\boldsymbol{X}$ is represented by d. The orientation error is denoted by $\tilde{\theta}$.

The orientation error $\tilde{\theta}$ can be computed with the following motor:

$$\boldsymbol{M}_h = 1 + \boldsymbol{L}_r \boldsymbol{L}_h, \tag{18.13}$$

and the lines $\boldsymbol{L}_h$ and $\boldsymbol{L}_r$, in the following way:

$$\widetilde{\theta} = \text{sign}(-(\boldsymbol{M} \cdot e_{12})\, \langle \boldsymbol{M} \rangle_0)\, \text{acos}(\boldsymbol{L}_h \cdot \boldsymbol{L}_r). \tag{18.14}$$

To compute the signed distance between the points $\boldsymbol{X}_o$ and $\boldsymbol{X}_r$, we use

$$d = s_d \sqrt{-\frac{1}{2} \boldsymbol{X}_o \cdot \boldsymbol{X}_r}, \tag{18.15}$$

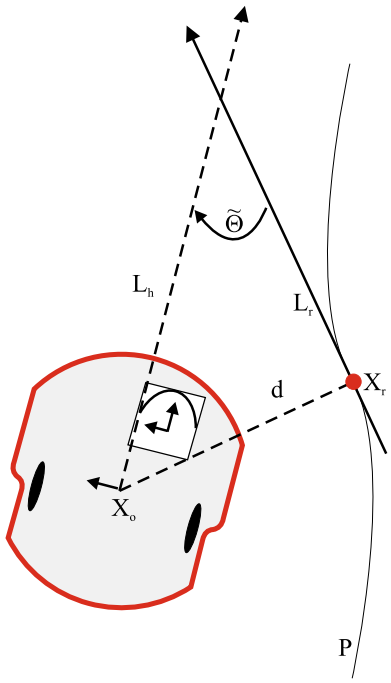


Fig. 18.11 The robot and the path to be followed

where s_d is defined as

$$s_d = \begin{cases} -1, & \boldsymbol{X}_o \cdot P_1 = 0, \\ 1, & else. \end{cases} \tag{18.16}$$

The point P_1 is obtained from the *point pair* Z, which is computed from the intersection of the circle C_r and the plane e_3 (Fig. 18.12). The circle C_r is defined using the line $\boldsymbol{L}_r$ and the point $\boldsymbol{X}_o$ with

$$C_r = \boldsymbol{X}_o \cdot \boldsymbol{L}_r. \tag{18.17}$$

Then, the *point pair* Z is computed with

$$Z = C_r^* \cdot e_3. \tag{18.18}$$

Finally, the point P_1 can be computed with

$$P_{1,2} = \frac{Z^* \pm |Z^*|}{Z^* \cdot e_\infty}\ . \tag{18.19}$$

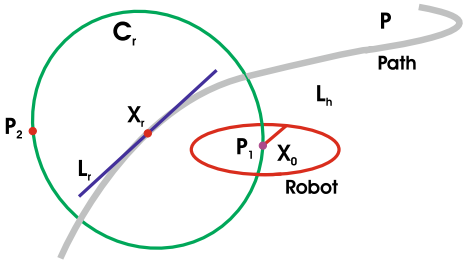


Fig. 18.12 Robot signed distance to the path

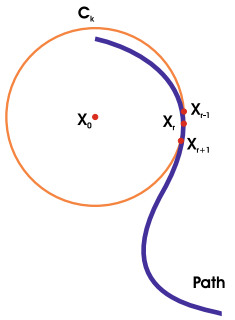


Fig. 18.13 How the curvature of the path can be computed using the circle C_k

In order to compute the path curvature c, we use the following circle (Fig. 18.13):

$$C_k = \boldsymbol{X}_{r-1} \wedge \boldsymbol{X}_r \wedge \boldsymbol{X}_{r+1}. \tag{18.20}$$

The circle radius can be computed with

$$r_k = \sqrt{\frac{-C_k^2}{(C_k \cdot e_\infty)^2}}. \tag{18.21}$$

Using the center point $\boldsymbol{X}_k$ of the circle C_k, we compute the line that passes through the points $\boldsymbol{X}_r$ and $\boldsymbol{X}_k$, that is,

$$\boldsymbol{L}_k = \boldsymbol{X}_r \wedge \boldsymbol{X}_k \wedge e_\infty. \tag{18.22}$$

The motor that transforms the line $\boldsymbol{L}_k$ into the line $\boldsymbol{L}_r$ is

$$\boldsymbol{M}_k = 1 + \boldsymbol{L}_r \boldsymbol{L}_k, \tag{18.23}$$

this motor yields the curvature sign

$$s_k = \text{sign}(-(\boldsymbol{M}_k \cdot e_{12})\, \langle \boldsymbol{M}_k \rangle_0). \tag{18.24}$$

Finally, the path curvature is defined with

$$c = s_k / r_k. \tag{18.25}$$

The controller used to generate the robot's angular velocity was proposed by Canudas et al. in [4] for path following, and shown to be stable:

$$\omega = \kappa_3 |v| \widetilde{\theta} - \kappa_2 v \, d \frac{\sin(\widetilde{\theta})}{\widetilde{\theta}} + \frac{v \cos(\widetilde{\theta}) c}{1 - c \, d}, \tag{18.26}$$

where κ_2 and κ_3 are constants to be tuned and c denotes the path curvature. The control gains are selected as

$$\kappa_2 = \alpha^2, \tag{18.27}$$

$$\kappa_3 = 2\xi\alpha, \tag{18.28}$$

where ξ is usually set to $1/\sqrt{2}$, meaning a small overshoot, and α is left free to specify faster or slower systems, i.e., shorter or longer settling times.

To describe more intuitively the control law (18.26), we analyze two particular important cases, namely, the cases where the reference path is a circle or straight line.

- When the reference path is a straight line, the curvature $c = 0$; therefore, the control is reduced to the simple case of encompassing only two terms. One term is proportional to the heading error and the other to the distance error.
- When the path is a circle, the curvature is constant. When the robot is close to the steady state, i.e., the heading and distance errors are close to zero, then the third term is constant. According to intuition, the robot must have a constant angular velocity to describe a circle.

The path-following task was tested in the global map the path was generated by hand giving some control points and interpolating between the lines or circles. Figure 18.14 has a sequence of images showing the path-following task. Figure 18.15 shows some pictures of the robot in the path-following task.

18.3 3D Map Building Using Laser and Stereo Vision

In this section, we present an interesting use of conformal geometric algebra to tackle problems of 3D map building and relocalization using a robot called "*geometer*" that is equipped with a laser rangefinder sensor and stereo camera system mounted on a pan–tilt head. These input devices have their own coordinate system, and so we apply *the method of hand–eye calibration* (see Sect. 16.1.2) to get the center coordinates of each device related to a global robot coordinate system. While *geometer* is moving

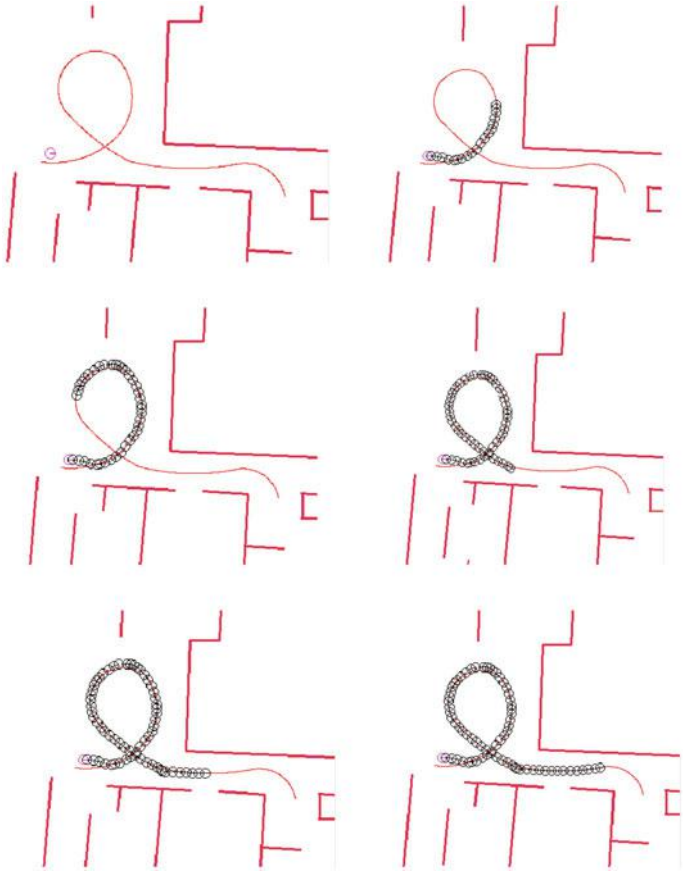


Fig. 18.14 The robot's initial pose in the map and the path to be followed

exploring the new areas, getting the data from laser rangefinder and the stereo camera system, two maps are performed simultaneously, one with local coordinates $\boldsymbol{L}$ (according to the current reading of the robot) and the other with global coordinates $\boldsymbol{G}$ (according to the initial position of exploration). The use of the encoders helps us to estimate the actual position of the mobile robot, but this has errors due to friction on the wheels. Therefore, the pose of the robot, its rotation angle, and translation are calculated by

$$\theta = \theta_o + \theta_{error}, \tag{18.29}$$

$$T = \boldsymbol{T}_o + \boldsymbol{T}_{error}, \tag{18.30}$$

where θ_o and T_o are the rotation angle and the translation vector given by the odometer, and θ_{error} and T_{error} are the value of correction error generated by the comparison

Fig. 18.15 Sequence of images showing the robot following the path

of the actual laser reading (line segments in local map) and the prior reading (line segments in global map). Using the perpendicular line to plane (x, y) as rotation axis and Eq. (18.29), and adding a third fixed coordinate to Eq. (18.30), we can apply these values in a translator $\boldsymbol{T}_{pos}$ and a rotor $\boldsymbol{R}_{pos}$ to represent the movement of the robot in the environment.

In the next section, we explain how to model data from the input devices in the 3D environment.

18.3.1 Laser Rangefinder

To extract line segments from range points, we use recursive line-splitting method shown in [5, 6], and this is a fast algorithm that performs *divide-and-conquer* algorithms [7]. For every endpoint of the line segments, we map them to CGA to get the pair of points entity (see Table 6.1) and store in a local map $\boldsymbol{L}$. As the endpoints are 2D point, we take the last coordinate in $\mathcal{V}^3$ and give the 0 value to fix the point in that plane. Now the local map $\boldsymbol{L}$ has every line segment represented as pair of points in CGA and we can apply any transformation on it (rotation, translation). While the map is being built, the collected data is stored in it with regard to the initial position. The following records taken from the laser rangefinder replace the actual local map for every new robot position in the environment. When a new local map $\boldsymbol{L}$ is taken, it is mapped to the global coordinate system using Eq. (18.34) to perform a line matching. Here we use one property of sphere to match line segments (pair of point), namely, having two spheres $\underline{s_1}^*$ and $\underline{s_1}^*$ the product

$$(\underline{s_1} \wedge \underline{s_2})^2 \begin{cases} < 0 \text{ if } \underline{s_1} \text{ and } \underline{s_2} \text{ intersect,} \\ = 0 \text{ if } \underline{s_1} \text{ and } \underline{s_2} \text{ are tangent,} \\ > 0 \text{ if } \underline{s_1} \text{ and } \underline{s_2} \text{ don't intersect,} \end{cases}$$

and to get a sphere from pair of points, we use

$$\underline{S_{PP^*}} = \frac{\underline{PP^*}}{\underline{PP^*} \wedge e_\infty}. \tag{18.31}$$

Figure 18.16 shows the possible combination of states of the line segments described below:

- a. Both spheres intersect and have the same direction.
- b. One sphere is inside of the other but has the same direction.
- c. Both spheres intersect but the segments do not have the same. direction.
- d. One sphere is inside of the other but the segments do not have the same direction.
- e. Both spheres do not intersect.

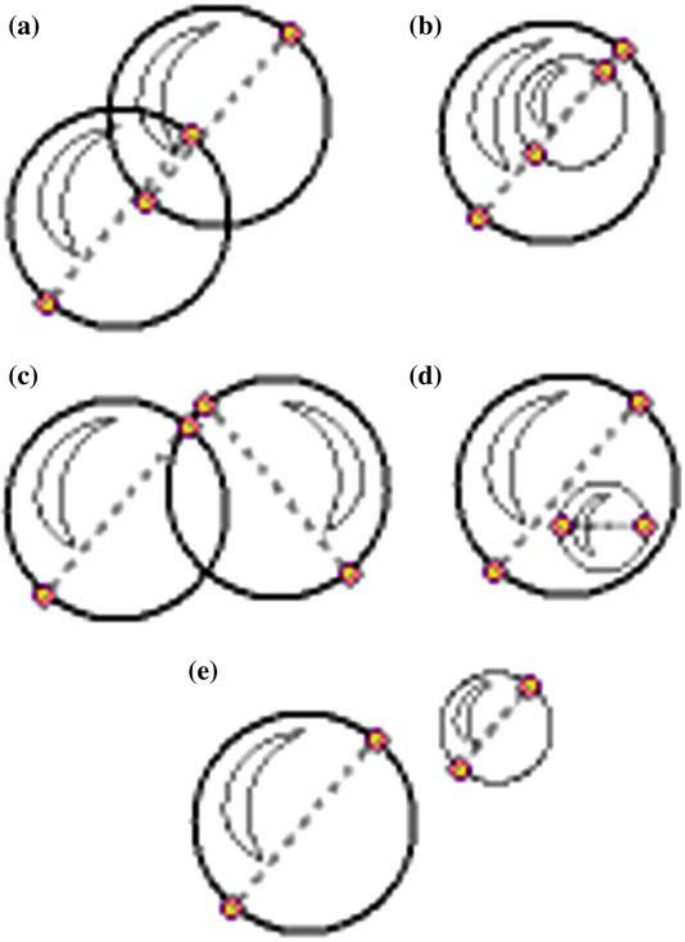


Fig. 18.16 Merging lines using spheres

When we get the lines matching, we merge both maps and correct the angle and displacement of the lines, comparing between local and global maps. This little error is caused by the odometry sensor. Then we update the actual position of the robot using Eqs. (18.29) and (18.30).

We can express a motor that maps any entity that has been taken from the laser coordinate systems to the global coordinate systems. We take as reference the laser's center and make a motor $\boldsymbol{M}_{lsr}$ that represents the rotation and translation from the center of the global coordinate systems to the laser's center and develop

$$\boldsymbol{M}_{cl} = \boldsymbol{R}_{pos}\boldsymbol{M}_{lsr}\widetilde{\boldsymbol{R}}_{pos}, \tag{18.32}$$

$$\boldsymbol{M}_{pos} = \boldsymbol{T}_{pos}\boldsymbol{R}_{lsr}, \tag{18.33}$$

$$\boldsymbol{M}_{lu} = \boldsymbol{M}_{cl}\boldsymbol{M}_{pos}, \tag{18.34}$$

where Eq. (18.32) is the translation and rotation motor toward laser's center, Eq. (18.33) is the movement of robot using laser rangefinder, and Eq. (18.34) is the motor that leads us to the source of the laser sensor in the global coordinate system (Fig. 18.17).

Using Eq. (18.34) with any geometric entity (points, lines, and circles) recorded with the laser rangefinder sensor, we can move easily to the global coordinate systems using the form

$$\underline{x}' = \boldsymbol{M}_{lu}\underline{x}\widetilde{\boldsymbol{M}}_{lu}. \tag{18.35}$$

As we are dealing in a 3D real world and the laser rangefinder only shows us a plane measure, we can add a virtual wall (Fig. 18.18) to the shapes from the laser rangefinder to get a 3D visual sense of the walls that are inside the virtual world.

If a new laser rangefinder is mounted on the mobile robot or if the laser rangefinder is moved to another place in the mobile robot, it is easy to get the new motor that maps the data from the laser rangefinder to the global map, only updating the motor $\boldsymbol{M}_{lsr}$ that represents the rotation and translation from the center of the global coordinate systems to the laser's center, and recalculate Eq. (18.34).

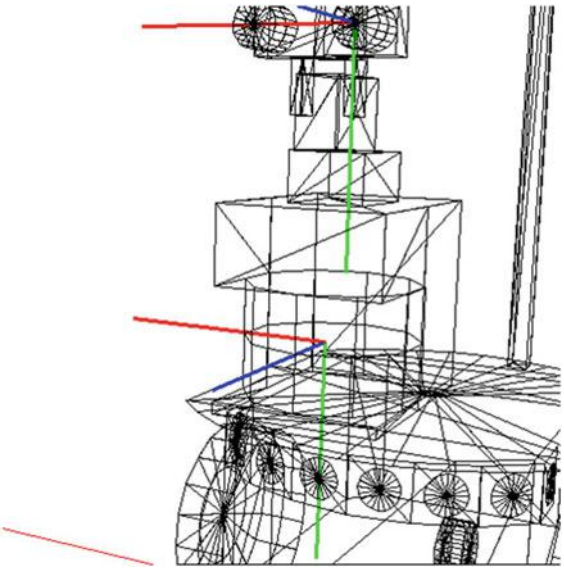


Fig. 18.17 Coordinate systems of the stereo pan–tilt unit and the laser rangefinder

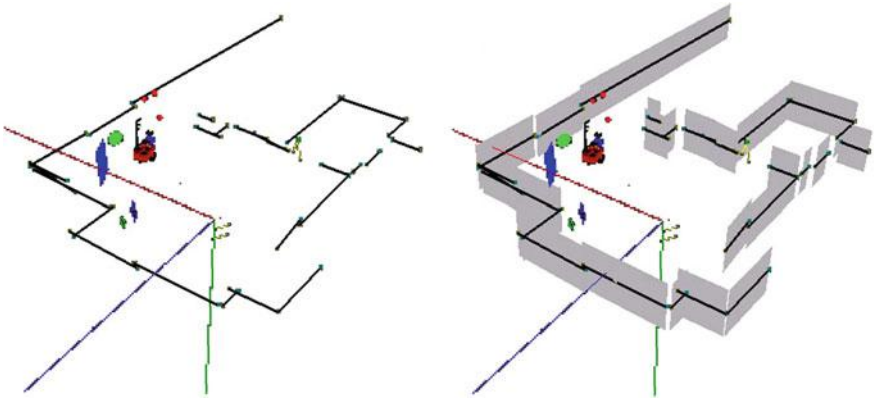


Fig. 18.18 3D map using virtual walls

18.3.2 Stereo Camera System with Pan–Tilt Unit

The pan–tilt unit has two degrees of freedom, which can be expressed as two rotations, one for pan and the other for tilt. This rotation can be modeled using rotors. Let $\boldsymbol{R}_{pan}$ be the rotor for the pan movement and $\boldsymbol{R}_{tilt}$ for the tilt movement. Applying these rotors using the geometric product, we can model the whole pan–tilt system. The stereo camera system has its center coordinates on the left camera (right camera viewing in front). We apply *the method of hand–eye calibration* [8] to get the axis from the pan–tilt unit getting its intersection (or the closet point between the rotation axis), and we build a translation from this intersection to the source of the stereo camera system. This translation is performed using a translator $\mathbf{T_{eye}}$. Now, with all this information, we develop a motor that maps any entities taken from the stereo camera system to the global coordinate systems as

$$\boldsymbol{T}_{ap} = \boldsymbol{R}_{pos}\boldsymbol{T}_{axis}\widetilde{\boldsymbol{R}}_{pos}, \tag{18.36}$$

$$\boldsymbol{R}_{pt} = \boldsymbol{R}_{pos}\boldsymbol{R}_{pan}\boldsymbol{R}_{tilt}, \tag{18.37}$$

$$\boldsymbol{T}_{opt} = \boldsymbol{R}_{pt}\boldsymbol{T}_{eye}\widetilde{\boldsymbol{R}}_{pt}, \tag{18.38}$$

$$\boldsymbol{M}_{mpt} = \boldsymbol{T}_{pos}\boldsymbol{R}_{pt}, \tag{18.39}$$

$$\boldsymbol{M}_{su} = \boldsymbol{T}_{opt}\boldsymbol{T}_{ap}\boldsymbol{R}_{mpt}, \tag{18.40}$$

where Eq. (18.36) is the translation to the point that has the minimum distance to the axis of the pan–tilt, taking into account the rotation of the robot position. Equation (18.37) is the rotor resulting from all the spins that have done so much in the position of the robot, as in the pan–tilt. Equation (18.38) is the translation to the left camera of the stereo camera system taking into account all the system's movements. Equation (18.39) is the movement motor of the robot, along with the pan–tilt. Equation (18.40) is the complete movement motor of the robot.

Any point captured by the cameras in any angle of the pan–tilt unit, in any position of the robot, can be mapped from the stereo camera system to the global coordinate systems using the form

$$\underline{x}' = \boldsymbol{M}_{su}\underline{x}\widetilde{\boldsymbol{M}}_{su}. \tag{18.41}$$

Using the CGA, we can capture all the entities shown in Table 6.1, using the OPNS form. By capturing the 3D objects using its representative points, we can represent points, line segments (pair of points), lines, circles, planes, and spheres in the frame of the stereo camera system and then take them to the global coordinate systems using (18.41).

18.4 Relocation Using Lines and the Hough Transform

The location of a robot in an environment that has already been captured is one of the problems that arise once the robot has finished the full map of its environment, and it is moved to an arbitrary place within it. The goal is to relocate the mobile robot within the map previously captured. This is known as the "kidnapping problem." The map that has been made has geometric information about the surrounding environment. From such information, we only use the records obtained by the laser rangefinder sensor, as lines are less noise sensitive.

The Hough transform [9] is a robust and effective method to identify the location and orientation of lines. The transform is the parametrization of a line from the (x, y) plane (a Cartesian plane) to the (θ, ρ) plane (the Hough domain). The line segments of the map are transformed to the Hough domain, defining the transformation in the domain of $\theta \in [0, 2\pi)$, so every line segment in (x, y) corresponds to a point (θ, ρ). This gives us one characteristic in a line, if it varies only in its angle θ, it keeps the value of ρ constant. So given a previously captured map G (global map) and a new captured map L (new local map), the difference between them is an angle $\Delta\theta$ and a displacement Δx and Δy that affects the ρ value.

$$\phi = G - L, \tag{18.42}$$

$$G' = \phi + L, \tag{18.43}$$

$$\sum G' - L = 0. \tag{18.44}$$

The difference of an angle in the Hough domain is defined as follows:

$$\Delta\theta(\theta_a, \theta_b) = \begin{cases} \theta_a - \theta_b - 2\pi & \text{if case 1,} \\ \theta_a - \theta_b + 2\pi & \text{if case 2,} \\ \theta_a - \theta_b & \text{otherwise,} \end{cases} \tag{18.45}$$

where

case 1 : if $\theta_a > \theta_b$, $\theta_a + \theta_\xi \geq 2\pi$, and $\theta_b + \theta_\xi \leq 0$,
case 2 : if $\theta_a < \theta_b$, $\theta_b + \theta_\xi \geq 2\pi$, and $\theta_a + \theta_\xi \leq 0$.

This gives us the calculus of any point near another where its angles are near 0 or 2π.

The relocation follows the following steps:

- Extract the actual environment, using the laser rangefinder, extract the line segment and map them to the Hough domain and store them in L (L only has (θ, ρ) from each line).
- Make the difference for each element in L with each element in G (using Eq. (18.45) in angles) and store it in $\Delta_{(\theta,\rho)}$, giving us a twist and displacement, this step can be seen as the difference between the actual map and the previously captured map.

$$\Delta_{(\theta,\rho)} = G - L. \tag{18.46}$$

- Now we build a new global map, adding all the elements of $\Delta_{(\theta,\rho)}$ to an element $l_i \in L$ and store it in G'_i as shown in Eq. (18.47):

$$G' = \Delta_{(\theta,\rho)} + L_i, \tag{18.47}$$

  which gives us a displacement of the actual map close to the global map.
- Now the angle $\Delta\theta$ have been shifted in G'_i, so we decrease G by the value of G'_i and get an error of displacement ξ_i. The goal is to reduce this error using

$$\sum G'_i - L = 0. \tag{18.48}$$

- Let V be a zero vote matrix of dimension $|G| \times |L|$, whose votes are given if the error of the displacement is less than a threshold

$$\xi_i < (\xi_\theta, \xi_\rho), \tag{18.49}$$

  where ξ_θ and ξ_ρ are the threshold from the angle and ρ, respectively.

Repeat last three steps for each line in L. Finally, when all the lines have been displaced and voted, extract the maximum value per column from V where the row position corresponds to a line in G, so this is the line correspondence if the value is null, there is no matching. Now we get all the information about which lines are matching, and with this data we can move and rotate the robot to the right place on the map according to the samples taken.

Use each matching line to get the average of the angles. With this angle, get the rotation angle to build a new *rotor* to turn the robot and the local map L (line segments) in the new environment. Now we have the orientation of the mobile robot, and we are only missing the displacement position. We can find the displacement

Δx and Δy using the closest point to the origin in the matching lines, to generate a translation vector. The closest point to the origin on a line in CGA can be calculated by

$$\mathbf{p} = -(\underline{L}^* \cdot \mathtt{E}) \cdot ((e_+ \cdot \underline{L})\mathtt{I}_\mathtt{E}). \tag{18.50}$$

As we get line segments (pair of points in CGA) we only need to apply the wedge operator with the point at infinity as shown in Eq. (18.51),

$$\underline{L}^* = \underline{PP}^* \wedge e_\infty, \tag{18.51}$$

to get the line in CGA and perform Eq. (18.50). With the translation vector we make a *translator* and apply it to the local map and to the mobile robot. And now that the robot is located in the correct place in the map, we can continue with navigation within the environment. In Fig. 18.19, we can see the relocation evolution, where (a) shows the initial position of the robot and is taking a sample of the environment, (b) generates the line segments of the actual environment, (c) loads the previous map to perform matching (here we can see that the mobile robot is displaced and turned in a random place), and (d) locates and puts the robot in the correct place in the map (here the robot has located itself on the previous environment and is placed in the right place).

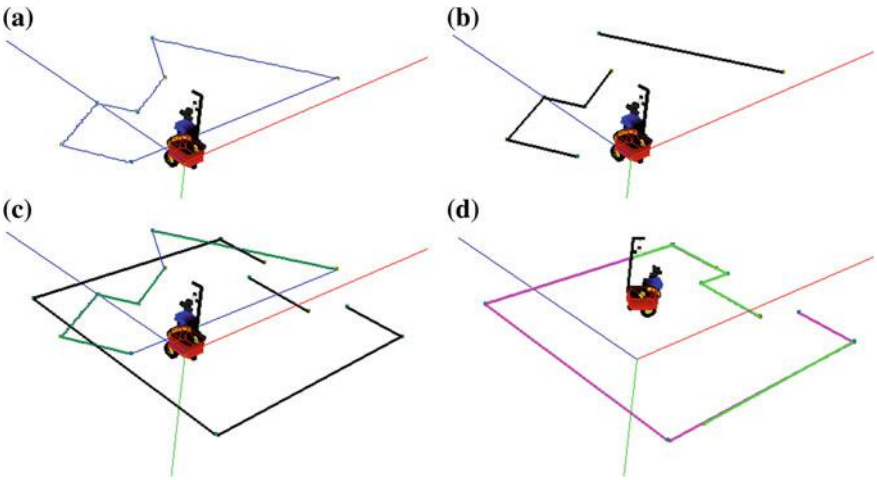


Fig. 18.19 Steps in relocation

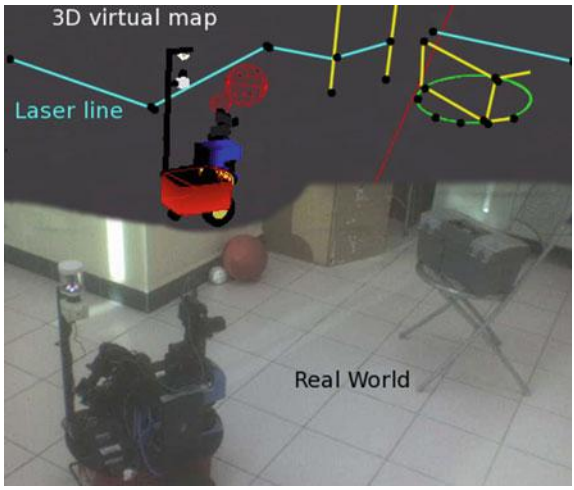


Fig. 18.20 Geometer registering the environment, merging laser and stereo data

18.5 Experiments

Figure 18.18 shows a virtual scenario using a laser rangefinder simulator and a real 3D object captured with the stereo camera. Here we can see that the 3D objects are in the place where the robot took the sample from them. Also we can enclose the taken environment using the virtual walls. These virtual walls help us to have the feeling of an indoor environment.

Figure 18.20 shows the creation of a real small 3D map merging the data registered by the laser rangefinder and the 3D shapes recorded by the stereo camera system. Here we can see that the laser line (line segments) is together with the rest of the 3D objects in the right pose.

All the maps are stored using only CGA entities, which help to reduce storage space. It has all the information about the objects inside the environment and will increment the map, not just continue with the last place where the last sample was taken. Also we can apply any transformation such as *rotation* or *translation* to continue with the map.

18.6 Conclusions

We have presented a complete theory for the solution of the body–eye calibration problem using motors of the conformal geometric algebra. The algorithm builds a matrix that can be solved using SVD, from which we get the solution motor. We also presented a scan-matching algorithm based on the body–eye calibration algorithm, which aligns the scans by representing the scan points as lines, which can be thought of as the screw axis of the body–eye calibration. We also present a path-following

task that also uses the conformal geometric algebra to estimate the geometric error for the control law. The chapter extended the applications of body-sensor calibration for the case of stereo vision and laser scanner. Such sensors are used for building 3D maps and tackling the relocalization problem. For the relocalization, we resort to an approach based on the Hough transform, where the desired position is searched in the line Hough space. The experiments with a real robot validate our method. Our approach can be of great use for mobile robots or upper body humanoids installed on moving platforms.

References

1. Hollerbach, J., & Nahvi, A. (1995). Total least squares in robot calibration. In *4th International Symposium on Experimental Robotics IV* (pp. 274–282).
2. Hartley, R. I., & Zissermann, A. (2003). *Multiple view geometry in computer vision* (2nd ed.). Cambridge: Cambridge University Press.
3. Bayro-Corrochano, E. (2010). *Geometric computing for wavelet transforms, robot vision, learning, control and action*. London: Springer Verlag.
4. Canudas, C., Siciliano, B., & Bastin, G. (1996). *Theory of robot control*. London: Springer-Verlag.
5. Zhang, L., & Ghosh, B. K. (2000). Line segment based map building and localization using 2D laser rangefinder. In *Proceedings of the IEEE International Conference on Robotics and Automation* (Vol. 3, pp. 2538–2543).
6. Siadat, A., Kaske, A., Klausmann, S., Dufaut, M., & Husson, R. (1997). An optimized segmentation method for a 2D laser-scanner applied to mobile robot navigation. In *Proceedings of the 3rd IFAC Symposium on Intelligent Components and Instruments for Control Applications* (pp. 153–158).
7. Nguyen, V., Gächter, S., Martinelli, A., Tomatis, N., & Siegwart, R. (2007). A comparison of line extraction algorithms using 2D range data for indoor mobile robotics. *Autonomous Robots*, *23*(2), 97–111.
8. Bayro-Corrochano, E., Daniilidis, K., & Sommer, G. (1997). Hand-eye calibration in terms of motions of lines using geometric algebra. In *10th Scandinavian Conference on Image Analysis* (Vol. I, pp. 397–404). Finland: Lappeenranta.
9. Hough, P. (1962). Methods and means for recognizing complex patterns. U.S. Patent 3 069 654.

Chapter 19
Quadrotor

19.1 Introduction

Unmanned autonomous vehicles, especially multi-copters, are becoming nowadays ubiquitous. Its popularity is due to its relative maneuverability in civil field for performing a wide range of applications, for example, monitoring roads or areas at risk, remote surveillance, inspection of power lines, etc. However, some of these applications require more specific control tasks, even more, robust controllers because external dynamics or disturbances, in general, affect the flight.

Most of the existing literature uses the approach of Newton–Euler formalism in order to obtain the dynamic model (e.g., [1]). Even though the dynamic equations of motion could be considered simple, it contains some nonlinearities and coupled terms which makes the design of a control law a challenging task. In addition, in most of the previous works, these equations are separated into the translational and rotational motions, and consequently, the control problem is divided into the position control problem and the attitude control problem, respectively. In the latter case, several representations of this dynamic have been discussed in the literature, such as Euler angles, Rodrigues parameters, axis/angle, and unit quaternion. Unit quaternion is a very efficient tool in attitude control due to its qualities, such as the use of the least possible number of parameters to represent orientation, its capacity to describe rotations in 3D space, its computational stability, and lack of the gimbal-lock effect inherent to Euler's angles.

In order to eliminate the need of designing two separate control for attitude and position, dual quaternion or motor algebra can be used because of its ability to deal in a most compact and efficient manner with attitude and position control problem simultaneously. There are some works using dual quaternion for multi-copters [2, 3], using a feedback control based on dual quaternion model, but external disturbances are not considered and the proposed control laws are designed only for stabilization of the system.

Recent works in control of multi-copters address the problem of minimizing disturbance in multi-copters [4, 5], using nonlinear control techniques such as sliding

© Springer Nature Switzerland AG 2020

E. Bayro-Corrochano, *Geometric Algebra Applications Vol. II*,
https://doi.org/10.1007/978-3-030-34978-3_19

mode in order to track a three-dimensional reference in the space. Nevertheless, as in most cases, the control problem is divided into position control and attitude control. Additionally, in that work a new method of adaptive function, due to internal dynamics, is presented.

In an attempt to merge both concepts, namely, unifying the dynamic model to deal with the position and attitude control simultaneously and the design of a control law which handles the tracking problem of a multi-copter in presence of large disturbances, in this work a block control technique [6] with second-order sliding mode [7] is proposed for trajectory tracking in presence of external disturbances and model uncertainties using a motor-algebra-based model.

19.2 Rigid Body Spatial Velocity Using Motor Algebra

In motor algebra, we represent the rigid motion of lines by Eq. (5.17). Lines are represented in terms of two dual bivector bases one for the line orientation and the second for the line moment. Considering the case of the pure rotor motion, we represent the relationship of the moving frames as follows:

$$l'_k(t) = \boldsymbol{M}(t) l_k \widetilde{\boldsymbol{M}}(t), \tag{19.1}$$

where l'_k and l_k are the coefficients of the two frames. Taking its time derivative

$$\dot{l}'_k = \dot{\boldsymbol{M}} l_k \widetilde{\boldsymbol{M}} + \boldsymbol{M} l_k \dot{\widetilde{\boldsymbol{M}}}, \tag{19.2}$$

and substituting $\boldsymbol{M} l_k = l'_k \boldsymbol{M}$, one gets

$$\dot{l}'_k = \dot{\boldsymbol{M}} \widetilde{\boldsymbol{M}} l'_k + l'_k \boldsymbol{M} \dot{\widetilde{\boldsymbol{M}}}. \tag{19.3}$$

Taking the time derivative of the identity $\boldsymbol{M}\widetilde{\boldsymbol{M}} = 1$, we get the useful relations

$$\begin{aligned}
V_s &= 2\dot{\boldsymbol{M}}\widetilde{\boldsymbol{M}} = 2(\dot{\boldsymbol{T}}\boldsymbol{R} + \boldsymbol{T}\dot{\boldsymbol{R}})\widetilde{\boldsymbol{R}}\widetilde{\boldsymbol{T}} = 2\dot{\boldsymbol{T}}\boldsymbol{R}\widetilde{\boldsymbol{R}}\widetilde{\boldsymbol{T}} + 2\boldsymbol{T}\dot{\boldsymbol{R}}\widetilde{\boldsymbol{R}}\widetilde{\boldsymbol{T}}, \\
&= 2\dot{\boldsymbol{T}}\widetilde{\boldsymbol{T}} + 2\Big(1 + I_m \frac{\boldsymbol{t}_s}{2}\Big)\Big(\frac{1}{2}\boldsymbol{\Omega}_s\Big)\Big(1 - I_m \frac{\boldsymbol{t}_s}{2}\Big), \\
&= I_m \dot{\boldsymbol{t}}_s \Big(1 - I \frac{\boldsymbol{t}_s}{2}\Big) + \Big(\boldsymbol{\Omega}_s + I_m \frac{\boldsymbol{t}_s}{2}\boldsymbol{\Omega}_s\Big)\Big(1 - I_m \frac{\boldsymbol{t}_s}{2}\Big), \\
&= I_m \dot{\boldsymbol{t}}_s + \Big(\boldsymbol{\Omega}_s + I_m \frac{\boldsymbol{t}_s}{2}\boldsymbol{\Omega}_s - I_m \boldsymbol{\Omega}_s \frac{\boldsymbol{t}_s}{2}\Big), \\
&= \boldsymbol{\Omega}_s + I_m \left(\dot{\boldsymbol{t}}_s + \frac{\boldsymbol{t}_s}{2}\boldsymbol{\Omega}_s - \boldsymbol{\Omega}_s \frac{\boldsymbol{t}_s}{2}\right), \\
&= \boldsymbol{\Omega}_s + I_m \Big(\dot{\boldsymbol{t}}_s + \frac{\boldsymbol{t}_s}{2} \underline{\mathrm{x}} \boldsymbol{\Omega}_s\Big) = \boldsymbol{\Omega}_s + I_m \Big(\boldsymbol{v}_s + \frac{\boldsymbol{t}_s}{2} \underline{\mathrm{x}} \boldsymbol{\Omega}_s\Big), \\
&= \boldsymbol{\Omega}_s + \mathrm{I}\mathcal{V}_s,
\end{aligned} \tag{19.4}$$

where the angular-velocity bivector $\boldsymbol{\Omega}_s$ is the dual of the linear-velocity bivector $\mathcal{V}_s = \boldsymbol{v}_s + \frac{t_s}{2}\underline{\mathrm{x}}\boldsymbol{\Omega}_s$.

Multiplying the Eq. (19.4) from the left by $\boldsymbol{M}$ and dividing by 2, we get the dynamic motor equation

$$\dot{\boldsymbol{M}} = \frac{1}{2}\boldsymbol{M}\boldsymbol{V}_S. \tag{19.5}$$

Assuming that the screw motion of the body is constant through time, the dynamic motor Eq. (19.5) can be integrated to give

$$\boldsymbol{M}(t) = e^{-\frac{\boldsymbol{V}_S}{2}}\boldsymbol{M}(0) = e^{-\frac{(\boldsymbol{\Omega}_S + I\boldsymbol{\mathcal{V}}_S)}{2}}\boldsymbol{M}(0). \tag{19.6}$$

This represents a motor that rotates with a constant-frequency rotation in the right-hand sense and has a constant linear velocity as well.

19.3 Mathematical Model Based on Motor Algebra to Multi-copter

Based on the Newton–Euler formalism and the forces transformed into the space of pure bivectors, it comes out

$$\begin{aligned}\sum F &= \frac{d}{dt}(m^{-1}\boldsymbol{v}_S) + \boldsymbol{\Omega}_S\underline{\mathrm{x}}(m^{-1}\boldsymbol{v}_S),\\ \sum T &= \frac{d}{dt}\mathcal{S}^{-1}(J^{-1}\mathcal{S}\{\boldsymbol{\Omega}_S\}) + J^{-1}\boldsymbol{\Omega}_S\underline{\mathrm{x}}\mathcal{S}^{-1}(J\mathcal{S}\{\boldsymbol{\Omega}_S\}),\end{aligned} \tag{19.7}$$

where $J = \mathrm{diag}(J_x,\ J_y,\ J_z)$ the inertia tensor, m the mass, and the transformation $\mathcal{S}$ relates the bivectorial magnitude and vector in $\mathbb{R}^3$ as follows $\boldsymbol{v}_S \in \mathbb{R}^3$, $\boldsymbol{v}_S = v_x e_1 + v_y e_2 + v_z e_3$, $\mathcal{S}\{\boldsymbol{v}_S\} = \boldsymbol{v}_S I_E$, and $I_E = e_1 e_2 e_3$ the Euclidean pseudoscalar. Besides, additional terms are expressed as follows:

$$\begin{aligned}\sum F &= F_{prop} - F_{aero} - F_{grav},\\ \sum T &= T_{prop} - T_{aero} - T_{gyro},\end{aligned} \tag{19.8}$$

where $T_{prop}, T_{aero}, T_{gyro} \in \mathbb{R}^3$ are aerodynamic and gyroscopic torques exerted by the propellers, respectively. F_{prop}, F_{aero} are bivector forces exerted by the propellers and aerodynamic forces, respectively. $F_{grav} = m\widetilde{R}_s G R_s$ is the force effected by gravity, with $G = (0, [0,\ 0,\ g]^T)m/s^2$ and the rotor R_s. These variables are defined as

$$
\begin{aligned}
F_{prop} &= \left(0, \begin{bmatrix} 0\,0 \sum_{i=1}^{n} F_i \end{bmatrix}^T\right), \\
T_{prop} &= \begin{bmatrix} d(\sum F_y^- - \sum F_y^+) \\ d(\sum F_x^- - \sum F_x^+) \\ c\sum_{i=1}^{n}(-1)^i F_i \end{bmatrix}, \\
T_{gyro} &= \sum_{i=1}^{n} J_R(\mathcal{S}\{\boldsymbol{\Omega}_S\} \times e_3^B)(-1)^{i+1}\omega_i,
\end{aligned} \tag{19.9}
$$

for n motors, where d is the distance from center of mass to rotor axis, c is the drag factor, J_R is the rotor inertia, ω_i is the i-th propeller velocity, $\sum F_y^-$, $\sum F_y^+$ are the sum of the breakdown along the y-axis negative and positive, respectively, analogously to the x-axis. Note that here the 6-vectors are represented in terms of the 6 bivectors of $G^+_{3,0,1}$, i.e., $e_{23}, e_{31}, e_{12}, e_{41}, e_{42}, e_{43}$.

The equations describing the dynamic multi-copter are obtained using (19.8) and (19.9) in (19.8), note that the centrifugal force $R_s[\boldsymbol{\Omega}_S \underline{\mathrm{x}}(m\boldsymbol{v}_S)]\widetilde{R}_s$ is nullified as the reference system of the earth is fixed

$$
\begin{aligned}
\mathcal{S}\{\dot{\boldsymbol{\Omega}}_S\} &= J^{-1}(T_{prop} - T_{aero} - T_{gyro} - \mathcal{S}\{\boldsymbol{\Omega}_S\} \times J\mathcal{S}\{\boldsymbol{\Omega}_S\}), \\
\dot{\boldsymbol{v}}_S &= m^{-1}R_s(F_{prop} - F_{aero})\widetilde{R}_s - G,
\end{aligned}
$$

then

$$
\dot{\boldsymbol{V}}_S = \widehat{F} + \widehat{U} + \widehat{w}, \tag{19.10}
$$

where

$$
\begin{aligned}
\widehat{F} &= \mathcal{S}^{-1}\{J^{-1}(-T_{gyro} - \mathcal{S}\{\boldsymbol{\Omega}_S\} \times J\mathcal{S}\{\boldsymbol{\Omega}_S\})\} + \\
&\quad + I_m\left[-G + \boldsymbol{v}_S \underline{\mathrm{x}} \boldsymbol{\Omega}_S + \right. \\
&\quad \left. + \boldsymbol{t}_S \underline{\mathrm{x}} \mathcal{S}\{J^{-1}(-T_{gyro} - \mathcal{S}\{\boldsymbol{\Omega}_S\} \times J\mathcal{S}\{\Omega_s\})\}\right], \\
\widehat{U} &= \mathcal{S}^{-1}\{J^{-1}T_{prop}\} + I_m\left[m^{-1}R_s F_{prop}\widetilde{R}_s + \right. , \\
&\quad \left. + \boldsymbol{t}_S \underline{\mathrm{x}} \mathcal{S}\{J^{-1}T_{prop}\}\right], \\
\widehat{w} &= \mathcal{S}^{-1}\{-J^{-1}T_{aero}\} + I_m\left[m^{-1}R_s(-F_{aero})\widetilde{R}_s\right. \\
&\quad \left. + \boldsymbol{t}_S \underline{\mathrm{x}} \mathcal{S}\{-J^{-1}T_{aero}\}\right],
\end{aligned} \tag{19.11}
$$

so that $\hat{w}$ is the aerodynamic wrench depending of the multi-copter velocity respect to air speed [8].

The model is described by (19.4) with state $M = R_s + I_m R_s'$ and (19.10) with state $\boldsymbol{V}_S = \boldsymbol{\Omega}_S + I_m \boldsymbol{\Omega}_S'$.

19.4 Control Design

The control objective is to track the output $\boldsymbol{t}_S = t_1 e_2 e_3 + t_2 e_3 e_1 + t_3 e_1 e_2$ and the orientation of multi-copter ψ, with the reference $\boldsymbol{t}_r, \psi_r$. Thus, the reference motor is $M_r = R_d(1 + \frac{1}{2}\boldsymbol{t}_r I_m)$, with $\boldsymbol{t}_r$ known and $R_d = r_{0d} + r_{1d} e_2 e_3 + r_{2d} e_3 e_1 + r_{3d} e_1 e_2$ as solution of the equation system

$$\begin{aligned} \psi_r &= tan^{-1}\left(\frac{2\,(r_{0d} r_{3d} + r_{1d} r_{2d})}{1 - 2(r_{2d}^2 + r_{3d}^2)}\right), \\ 1 &= r_{0d}^2 + r_{1d}^2 + r_{2d}^2 + r_{3d}^2, \end{aligned} \tag{19.12}$$

and other three equations deduced later.

The tracking error is defined as $M_e = M_r \widetilde{M}$. With origin $O = 1 + 0 I_m = 1$, and take asymptotically to zero $\hat{z}_1 = \log(M_e) = z_1 + I_m z_1'$. One works out the error dynamics $\hat{z}_1$ results

$$\begin{aligned} \hat{z}_1 &= \log(R_d \widetilde{R}_s) + (\boldsymbol{t}_r - \boldsymbol{t}_S) I_m, \\ \dot{\hat{z}}_1 &= \frac{d}{dt}\log(R_d) - 1/2\ \Omega_s + I_m[\dot{\boldsymbol{t}}_r - \Omega_T + \Omega_s \underline{\mathrm{x}}(\widetilde{R}_s R_s')], \end{aligned}$$

where according to Eq. (4.52) $\boldsymbol{t}_S = 2\widetilde{R}_s R_s'$. It should be noted that the derivative of reference rotor logarithm is $d/dt\,\log(R_d) = \mathcal{S}^{-1}\{B_2 \mathcal{S}\{\Omega_s\} + \bar{f}\}$, where the matrix B_2 and the vector $\bar{f}$ are unknown functions.

The following assumption is instrumental to get the desired result.

Assumption 1 The unknown matrix B_2 and vector $\bar{f}$ are bounded and can be expressed as Kronecker series dependent on the state $R_s = r_0 + r_1 e_2 e_3 + r_2 e_3 e_1 + r_3 e_1 e_2$, this is

$$B_{23} = \sum_{l=0}^{\infty} \frac{1}{l!} C_l I_3 \otimes \mathcal{S}\{R_s^l\}, \tag{19.13}$$

$$\bar{f} = \sum_{l=0}^{\infty} \frac{1}{l!} D_l \mathcal{S}\{R_s^l\}, \tag{19.14}$$

where $C_l \in I\!R^{3\times 3(4^l)}$ and $D_l \in I\!R^{3\times 4^l}$ are constants, and I_3 is the identity matrix of order 3. Besides $R_s^l = R_s^{l-1} R_s$ with $R_s^0 = 1$ and the operator $\otimes$ as the Kronecker product.

Then an adaptive estimator for $\log(R_d)$ is defined as

$$\dot{\hat{\gamma}} = \widehat{B}_2 \mathcal{S}\{\Omega_s\} + \widehat{\bar{f}} + \epsilon, \tag{19.15}$$

where

$$\epsilon = \alpha_1 \tilde{\gamma} + \alpha_2 \mathrm{sign}\,(\tilde{\gamma}) \tag{19.16}$$

is used to ensure stability for $\tilde{\gamma} = \mathcal{S}\{\log(R_d)\} - \hat{\gamma}$ and

$$\widehat{B}_2 = \sum_{l=0}^{N} \frac{1}{l!} \widehat{C}_l I_3 \otimes \mathcal{S}\{R_s^l\},$$
$$\widehat{\tilde{f}} = \sum_{l=0}^{N} \frac{1}{l!} \widehat{D}_l \mathcal{S}\{R_s^l\} \tag{19.17}$$

are the estimate functions, with $\widehat{C}_l$ and $\widehat{D}_l$ as constants to be adapt with dynamic

$$\dot{\widehat{C}}_l = \frac{\gamma_2}{l!} \tilde{\gamma} \mathcal{S}\{\Omega_s\}^T (I_3 \otimes \mathcal{S}\{R_s^l\})^T,$$
$$\dot{\widehat{D}}_l = \frac{\gamma_1}{l!} \tilde{\gamma} \mathcal{S}\{R_s^l\}^T. \tag{19.18}$$

The error dynamics become

$$\dot{\tilde{\gamma}} = \widetilde{B}_2 \mathcal{S}\{\Omega_s\} + \widetilde{\tilde{f}} + \bar{e} - \epsilon, \tag{19.19}$$

with

$$\widetilde{B}_2 = \sum_{l=0}^{N} \frac{1}{l!} \widetilde{C}_l I_3 \otimes \mathcal{S}\{R_s^l\},$$
$$\widetilde{\tilde{f}} = \sum_{l=0}^{N} \frac{1}{l!} \widetilde{D}_l \mathcal{S}\{R_s^l\},$$
$$\bar{e} = \sum_{l=N+1}^{\infty} \frac{1}{l!} C_l I_3 \otimes \mathcal{S}\{R_s^l\} \mathcal{S}\{\boldsymbol{\Omega}_s\} + \sum_{l=N+1}^{\infty} \frac{1}{l!} D_l \mathcal{S}\{R_s^l\}$$

and the adaptive errors

$$\widetilde{C}_l = C_l - \widehat{C}_l,$$
$$\widetilde{D}_l = D_l - \widehat{D}_l.$$

The following result shows the convergence of the error to the equilibrium point.

Theorem 19.4.1 *Let Assumption 1 hold and consider the adaptive estimator* (19.15), (19.18) *for the system*

$$\mathcal{S}\{d/dt \ \log(R_d)\} = B_2 \mathcal{S}\{\boldsymbol{\Omega}_s\} + \bar{f}.$$

Then the equilibrium point $\tilde{\gamma} = 0$ *of the closed-loop system* (19.19), (19.16) *is asymptotically stable.*

Proof Let a Lyapunov function

$$\mathcal{V}_\gamma = \frac{1}{2}\tilde{\gamma}^T\tilde{\gamma} + \frac{1}{2\gamma_2}\sum_{l=0}^{\infty}\sum_{r=1}^{3}\sum_{t=1}^{3(4^l)}\widetilde{c}_{rtl}^2 + \frac{1}{2\gamma_1}\sum_{l=0}^{\infty}\sum_{r=1}^{3}\sum_{t=1}^{4^l}\widetilde{d}_{rtl}^2, \tag{19.20}$$

where the terms $\widetilde{c}_{rtl}$ and $\widetilde{d}_{rtl}$ are the element r, t of the matrix $\widetilde{C}_l$ and $\widetilde{D}_l$, respectively. From (19.20) and (19.19), one obtains

$$\begin{aligned}\dot{\mathcal{V}}_\gamma = &-\tilde{\gamma}^T\epsilon + \tilde{\gamma}^T\left(\sum_{k=0}^{N}\frac{1}{l!}\widetilde{C}_l I_3 \otimes \mathcal{S}\{R_s^l\}\right)\mathcal{S}\{\boldsymbol{\Omega}_s\}+\\ &+\frac{1}{\gamma_2}\sum_{l=0}^{N}\sum_{r=1}^{3}\sum_{t=1}^{3(4^l)}\widetilde{c}_{rtl}\dot{\widetilde{c}}_{rtl}+\\ &+\sum_{l=0}^{N}\left(\tilde{\gamma}^T\frac{1}{l!}D_l\mathcal{S}\{R_s^l\} + \frac{1}{\gamma_1}\sum_{r=1}^{3}\sum_{t=1}^{4^l}\widetilde{d}_{rtl}\dot{\widetilde{d}}_{rtl}\right) + \tilde{\gamma}^T\bar{e},\end{aligned}$$

notice that $\dot{\widetilde{c}}_{rtl} = -\dot{\widehat{c}}_{rtl}$ and $\dot{\widetilde{d}}_{rtl} = -\dot{\widehat{d}}_{rtl}$. Then, from Eq. (19.18), this results

$$\dot{\mathcal{V}}_\gamma = -\tilde{\gamma}^T\epsilon + \tilde{\gamma}^T\bar{e}.$$

Since B_2 and $\bar{f}$ are bounded, then

$$\|\bar{e}\| \leq \delta_1\|\tilde{\gamma}\| + \delta_2$$

is also bounded. So if $v = \alpha_1\tilde{\gamma} + \alpha_2\text{sign}(\tilde{\gamma})$, it follows that

$$\begin{aligned}\dot{\mathcal{V}}_\gamma &= -\tilde{\gamma}^T\alpha_1\tilde{\gamma} - \tilde{\gamma}^T\alpha_2\text{sign}(\tilde{\gamma}) + \bar{e},\\ &\leq -\alpha_1\|\tilde{\gamma}\|^2 - \alpha_2\|\tilde{\gamma}\| + \delta_1\|\tilde{\gamma}\|^2 + \delta_2\|\tilde{\gamma}\|,\end{aligned} \tag{19.21}$$

so that $\alpha_1 > \delta_1$ and $\alpha_2 > \delta_2$ make the system (19.19) asymptotically stable. Also, using Barbalat's lemma and LaSalle's theorem, stability in subsystem (19.18) is also ensured.

Now, from Theorem 19.4.1, the closed-loop system (19.19) with the adaptable law (19.18) has a stable equilibrium point $\tilde{\gamma} = 0$. Then the desired states are proposed as pseudo-control Ω as shown in [6], this is

$$\begin{aligned}\boldsymbol{V}_d &= \boldsymbol{\Omega}_{sd} + I_m\boldsymbol{\Omega}'_{sd},\\ \boldsymbol{\Omega}_{sd} &= 2\mathcal{S}^{-1}\left\{\left(\widehat{B}_2 - 1/2\ I_3\right)^{-1}\left(\widehat{\bar{f}} - 2k_{1R}\mathcal{S}\{z_1\}\right)\right\},\\ \boldsymbol{\Omega}'_{sd} &= \dot{\boldsymbol{t}}_r + 2\boldsymbol{\Omega}_{sd}\underline{\mathrm{x}}(\widetilde{R}_s R'_s) + \mathcal{S}^{-1}\{k_{1T}\mathcal{S}\{z'_1\}\},\end{aligned}$$

where k_{1q} and k_{1T} are the gain matrices control to stabilize $\hat{z}_1$. Now, defining $\hat{z}_2 = \boldsymbol{V}_d - \boldsymbol{V}_S$, results

$$\dot{\hat{z}}_2 = \dot{\boldsymbol{V}}_d - \widehat{F} - \widehat{U} - \widehat{w}.$$

Otherwise, let define $\dot{\hat{z}}_2 = z_2 + z_{2T}$, then

$$\dot{\hat{z}}_2 = \dot{\boldsymbol{\Omega}}_{sd} - U_T + I_m(\dot{\boldsymbol{\Omega}}'_{sd} - m^{-1} R_d F_{prop} \widetilde{R}_d + U_T \underline{\mathrm{x}} \boldsymbol{t}_S) - \hat{F},$$

therefore $\hat{z}_2$ is chosen as sliding surface. The control is designed through the super-twisting algorithm [7]

$$\begin{aligned} \widehat{U} &= \dot{\widehat{\boldsymbol{\Omega}}}_d - \widehat{F} + k_{2R} \|z_2\| \operatorname{sign}(z_2)) + I_m k_{2T} \left\|z'_2\right\| \operatorname{sign}\left(z'_2\right) - \widehat{V}, \\ \dot{\widehat{V}} &= -\alpha_R \operatorname{sign}(z_2) - I_m \alpha_T \operatorname{sign}\left(z'_2\right), \end{aligned} \tag{19.22}$$

with $\widehat{U} = U_T + I_m U_F$, also $k_{2R}, k_{2T}, \alpha_R, \alpha_T$ are scalars to stabilize the sliding surface $\hat{z}_2 = z_2 + I_m z'_2$.

Finally, we have

$$\begin{aligned} m^{-1} R_d F_{prop} \widetilde{R}_d = U_F - I_m U_T \underline{\mathrm{x}} \boldsymbol{t}_S = m^{-1}(2(r_{1d} r_{3d} + r_{0d} r_{2d}) e_{23} + \\ 2(r_{0d} r_{1d} + r_{2d} r_{3d}) e_{31} + (r_{0d}^2 - r_{1d}^2 - r_{2d}^2 + r_{3d}^2) e_{12}) \sum_i^n F_i, \end{aligned}$$

where $R_d = r_{0d} + r_{1d} e_{23} + r_{2d} e_{31} + r_{3d} e_{12}$ and $T_{prop} = J U_T$, which, in conjunction with Eq. (19.12), solves the system of nonlinear equations for R_d and for the forces of each motor F_i.

To show the stability of the closed-loop system (19.4), (19.10), (19.22), (19.15), (19.16), (19.18) suppose that the external perturbation is bounded by $\|\widehat{w}\| \leq b \left\|\hat{z}_2\right\|^{1/2}$, where the closed-loop system is

$$\begin{aligned} \dot{\hat{z}}_1 &= -(1/2 I_3 + \widehat{B}_2)^{-1}(k_{1R} \mathcal{S}\{z_1\} + k_{1T} \mathcal{S}\{z'_1\}) + \widetilde{\bar{f}} + \widetilde{B}_2 \mathcal{S}\{\boldsymbol{\Omega}_s\} - \hat{z}_2, \\ \dot{\hat{z}}_2 &= STA(\hat{z}_2) + \widehat{w}, \end{aligned}$$

where $STA(\hat{z}_2)$ is the super-twisting algorithm applying to $\hat{z}_2$ as in (19.11). After the system enters the sliding regime, we obtain $\hat{z}_2 = 0$, and thus consider the Lyapunov function

$$\mathcal{V} = \frac{1}{2}(\|z_1\|^2 + \left\|z'_1\right\|^2) + \mathcal{V}_\gamma.$$

Then, we get

$$\begin{aligned} \dot{\mathcal{V}} = -\mathcal{S}\{z_1\}^T (1/2 I + \widehat{B}_2)^{-1}(k_{1R} \mathcal{S}\{z_1\} + k_{1T} \mathcal{S}\{z'_1\}) + \\ + \mathcal{S}\{z_1 + z'_1\}^T \left(\widetilde{\bar{f}} + \widetilde{B}_2 \mathcal{S}\{\Omega_s\}\right) - (\alpha_1 - \delta_1) \|\tilde{\gamma}\|^2 - (\alpha_2 - \delta_2) \|\tilde{\gamma}\|. \end{aligned} \tag{19.23}$$

Assuming that the estimation error for $\widetilde{\widetilde{f}} + \widetilde{B}_2 \mathcal{S}\{\boldsymbol{\Omega}_s\}$ is bounded as $\widetilde{\widetilde{f}} + \widetilde{B}_2 \mathcal{S}\{\boldsymbol{\Omega}_s\} \leq \delta \|z_1\|$, it comes that

$$\dot{\mathcal{V}} \leq -(\lambda_{min}(k_{1R}, k_{1T}) - \delta) \left\|\hat{z}_1\right\|^2 - (\alpha_2 - \delta_2) \|\tilde{\gamma}\| - (\alpha_1 - \delta_1 - \delta) \|\tilde{\gamma}\|^2 .$$

Using (19.20), (19.24) we finally obtain

$$\dot{\mathcal{V}} \leq -\theta \left\|\hat{z}_1\right\| - (\alpha_1 - \delta_1) \|\tilde{\gamma}\| - (\alpha_2 - \delta_2) \|\tilde{\gamma}\|^2$$

with θ as positive constant. Thus, the Lyapunov function is negative definite, and thus the solution $\left\|\hat{z}_1(t)\right\|$ tends asymptotically to zero.

19.5 Simulation Results

To verify the effectiveness and robustness of the proposed feedback control law detailed above, a simulation of a quadrotor flight system is presented. In order to apply this control law, it is necessary to compute the angular velocities ω_i of the motors, which are obtained from the approximation of $F_i = b\omega_i^2$ to i-th actuator, with b as the thrust factor, $F_i = c\omega_i^2$ for the tangential force, and c as the drag factor. It follows that

$$\begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{bmatrix} = \begin{bmatrix} 0 & -db & 0 & db \\ -db & 0 & db & -db \\ -c & c & -c & c \\ b & b & b & b \end{bmatrix}^{-1} u,$$

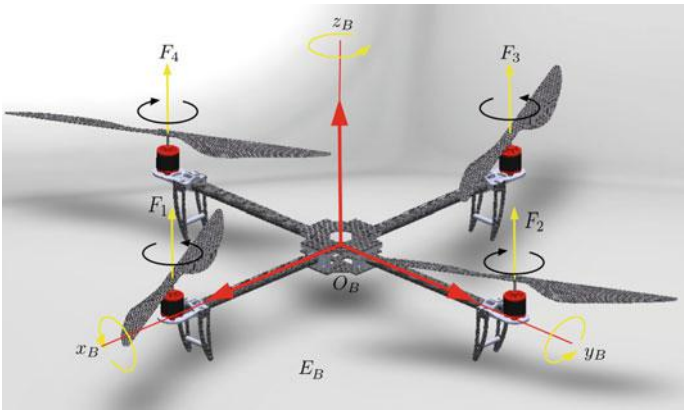


Fig. 19.1 Forces and movements of a quadrotor

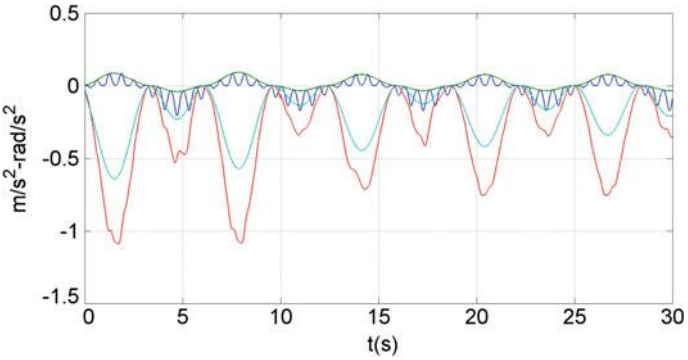


Fig. 19.2 Aerodynamic acceleration

where d is the distance from center of mass to rotor axis, ω_i the i-th propeller velocity for $i = \{1, 2, 3, 4\}$ as defined previously, and $u = [F_1, F_2, F_3, F_4]^T$. Figure 19.1 depicts the forces and the movements of the quadrotor.

The reference for the linear position is formulated by using points in the following coordinates (x_r, y_r, z_r) in an earth reference frame which are given by the points (0, 0, 1), (1, 0, 1), (1, 1, 1), (0, 1, 1), and (0, 0, 1) in meters at intervals of 4 s. Table 19.1 shows the parameters of the quadrotor used in the simulation.

To demonstrate the robustness of the proposed control law, the mass m and the inertial tensor I are increased by 30% of the initial value at $t = 10$ s. In addition, external disturbances as aerodynamic accelerations are included as shown in Fig. 19.2.

Figures 19.3 and 19.4 depict the change of position and the angles of the quadrotor, respectively. It can be seen that at $t = 10$ s the system is perturbed causing a small transitory behavior in both translational and rotational motions due to parametric variation; however, the control laws manage to keep the tracking of the reference despite these transients and external disturbances.

Figure 19.5 displays the position in the earth reference frame. It can be observed that the quadrotor tracks satisfactorily the given reference. The slight deviation next

Table 19.1 Parameters of quadrotor

Parameter	Value
m	4.900
$I_{x,y}$	21.6×10^{-3}
I_z	43.2×10^{-3}
J_R	3.357×10^{-5}
d	0.45
c	1.140×10^{-7}
b	2.98×10^{-5}

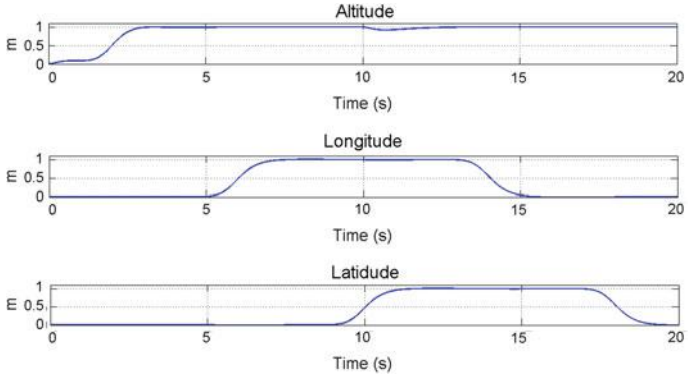


Fig. 19.3 Translational motion

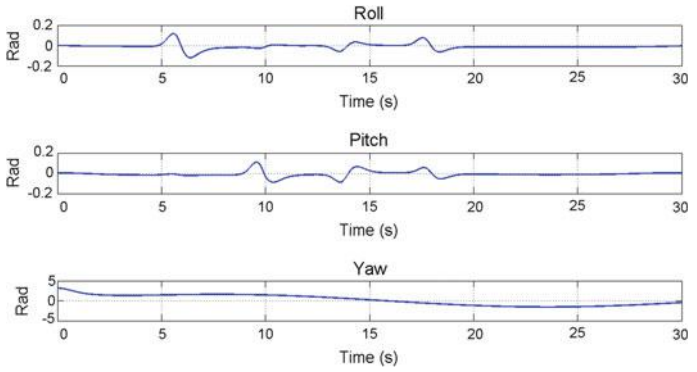


Fig. 19.4 Rotational motion (angles)

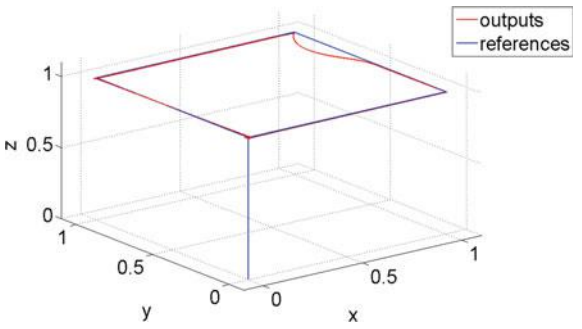


Fig. 19.5 Trajectory in earth reference frame

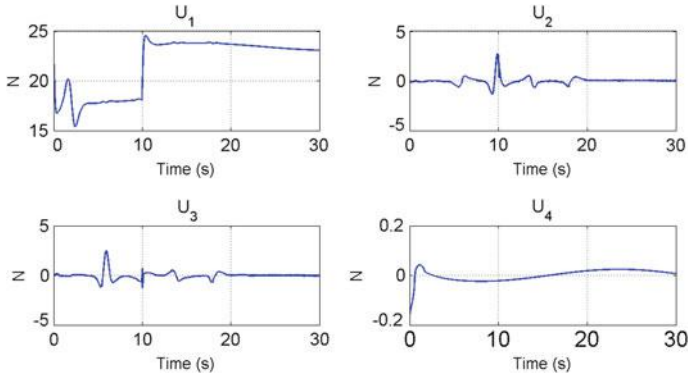


Fig. 19.6 Control law

to the third corner is due to the added perturbation at $t = 10$ s, but as we see the control manages to return to the expected trajectory.

Finally, the control signal is presented in Fig. 19.6, which has a smooth behavior and it is within the range allowed by the actuators despite the external disturbances and parametric variations. It is shown at $t = 10$ s, how the control laws react under the added perturbation.

19.6 Conclusion

In this work, the dynamic model and the nonlinear control for a multi-copter have been developed using the geometric algebra framework specifically using the motor algebra $G^{+}_{3,0,1}$. The kinematic for the aircraft model and the dynamic based on Newton–Euler formalism are presented and the block control technique is applied in combination with super-twisting including an internal dynamics estimator driven by maneuvers away from the origin. Finally, the stability of the presented control scheme is shown.

It has been also shown that our nonlinear controller law is able to reject external disturbances and can deal with parametric variations. This control has the computational advantages inherited from geometric algebra compared to the classical use of Euler angles. Specifically, motor algebra is used due to its excellent ability to deal with the need of two separate control laws for attitude and position, allowing to solve both problems simultaneously in a more compact and efficient manner.

References

1. Bouabdallah, S., & Siegwart, R. (2007). Design and control of an indoor micro quadrotor. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, San Diego, CA, USA.

2. Wang, X., & Yu, C. (2010). Feedback linearization regulator with coupled attitude and translation dynamics based on unit dual quaternion. In *IEEE International Symposium on Intelligent Control* (pp. 2380–2384).
3. Abaunza, H., Cariño, J., Castillo, P., & Lozano, R. (2015). Quadrotor dual quaternion control. In *2015 Workshop on Research, Education and Development of Unmanned Aerial Systems (RED-UAS)* (pp. 195–203).
4. Luque-Vega, L., Castillo-Toledo, B., & Loukianov, A. G. (2012). Robust block second order sliding mode control for a quadrotor. Advances in guidance and control of aerospace vehicles using sliding mode control and observation techniques. *Journal of the Franklin Institute, 349*, 719–739
5. Arellano-Muro, C. A., Castillo-Toledo, B., Loukianov, A. G., Luque-Vega, L. F., & Gonzalez-Jimenez, L. (2015). Quaternion-based trajectory tracking robust control for a quadrotor. In *10th Annual System of Systems Engineering Conference 2015 (SoSE 2015)*, San Antonio, USA, 17 May 2015.
6. Loukianov, A. G. (2002). Robust block decomposition sliding mode control design. *International Journal Mathematical Problems in Engineering: Theory, Methods and Application* 349–365.
7. Levant, A. (2005). Universal single-input-single-output (SISO) sliding-mode controllers with finite time convergence. *IEEE Transactions on Automatic Control* 1785–1789.
8. Gessow, A., & Myers, G. C. (1978). *Aerodynamics of the helicopter* Fifth printing. New York: Frederick Unger Publishing Co.

Part V
Applications II: Medical Robotics

Chapter 20
Modeling and Registration of Medical Data

20.1 Background

In medical image analysis, the availability of 3D models is of great interest to physicians because it allows them to have a better understanding of the situation, and such models are relatively easy to build. However, sometimes and in special situations (such as surgical procedures), some structures (such as the brain or tumors) suffer a (nonrigid) transformation and the initial model must be corrected to reflect the actual shape of the object. In the literature, we can find the Union of Spheres algorithm [1], which uses the spheres to build 3D models of objects and to align or transform it over time. In our approach, we also use the spheres, but we use the marching cubes algorithm's ideas to develop an alternative method, which has the advantage of reducing the number of primitives needed; we call our method *marching spheres*.

20.1.1 Union of Spheres

This algorithm was proposed in [1] and can be summarized as follows:

- Given a set of boundary points (borders of the 3D volumetric data), calculate the Delaunay Tetrahedralization (DT).
- Compute the circumscribing sphere to each tetrahedron.
- Verify from the original data which spheres are inside the defined object.
- Simplify the dense sphere representation by clustering and eliminating redundant or nonsignificant spheres.

This algorithm has the worst-case complexity of $O(n^2)$ in both time and number of primitives, where n is the number of boundary points. However, in [1], it is noted that the highest number of primitives observed in experiments was $4n$, approximately. This number, although good, could be computationally heavy for big n.

© Springer Nature Switzerland AG 2020

E. Bayro-Corrochano, *Geometric Algebra Applications Vol. II*,
https://doi.org/10.1007/978-3-030-34978-3_20

To register models based on spheres, the authors first match a sufficient number of spheres from the Union of Spheres representation of one object to the other, and then from the matches find the most likely transformation using some methods like least squares.

20.1.2 The Marching Cubes Algorithm

The basis of the marching cubes algorithm is to subdivide the space into a series of small cubes. The algorithm then moves (or "marches") through each of the cube testing the corner points to determine whether or not they are on/inside the surface and replacing the cube with an appropriate set of polygons. The resulting set of polygons will be a surface that approximates the original one. To explain the algorithm, let us look at a two-dimensional equivalent. Suppose we want to approximate a 2D shape like the one in Fig. 20.1a; then we create a grid of squares (equivalent to the cubes for the 3D version of the algorithm). The first task is to determine which of the corners of these squares are inside (or on) the shape (see Fig. 20.1b). Then we insert new vertices that are positioned halfway between each inside and outside corner connected to each other by an edge of the grid (see Fig. 20.1c). Finally, we join the inserted vertices using lines and obtain an approximation to the original surface (Fig. 20.1d). Obviously, the finer the grid, the better the approximation to the shape.

For the 3D case, suppose we have a set of m images (slides) containing the shape information of the object we want to represent. Then we create a set of *logical cubes*, with each cube created from eight pixels: four of slide k and four of slide $k + 1$. The algorithm determines how the cube intersects the surface and then moves (or marches) to the next cube. At each cube, we assign a one to a cube's vertex if this vertex is inside (or on) the surface and a zero if this vertex is outside the surface.

Since there are weight vertices in each cube and two states for each one, there are $2^8 = 256$ ways the surface can intersect the cube. In [2], it is shown how these 256 cases can be reduced to only 15 for clarity in explanation, although a permutation of these 15 cases using complementary and rotational symmetry produces the 256 cases. With this number of basic cases, it is then easy to create predefined polygon sets for making the appropriate surface approximation.

Before we proceed to explain our proposed method for volume representation and registration based on spheres, let us introduce the basic concepts of geometric algebra and conformal geometric algebra, because we will use the representation of spheres in such algebras for such tasks.

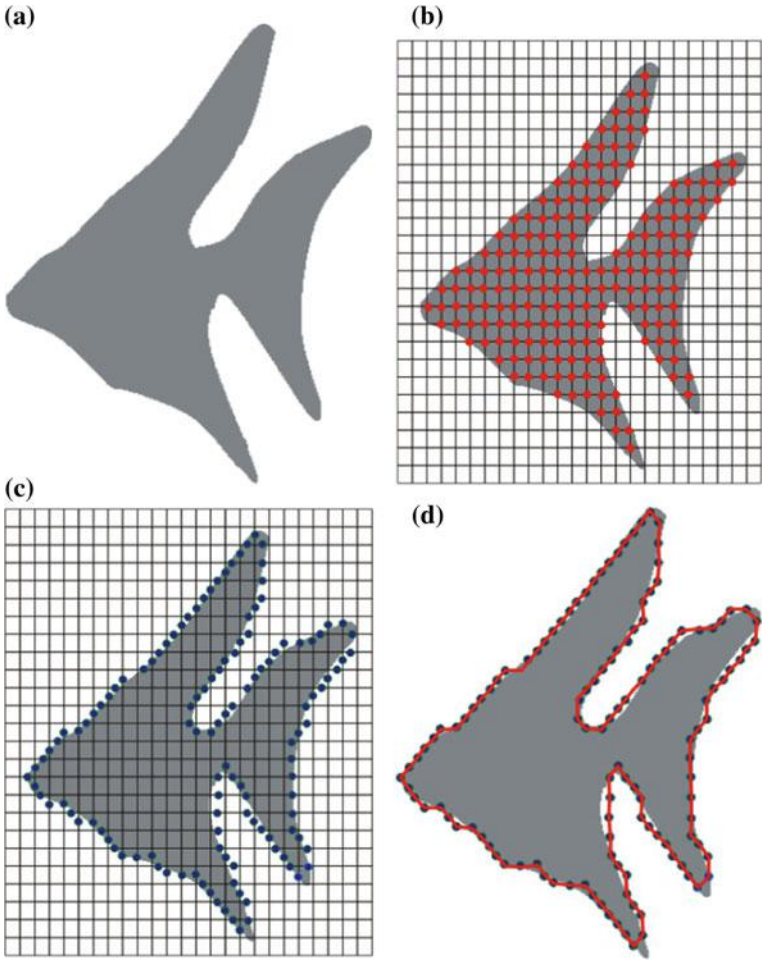


Fig. 20.1 The 2D equivalent of marching cubes. **a** The shape we want to represent; **b** determining which corners of the grid are inside the shape; **c** new vertices inserted; **d** approximation to the surface by joining the vertices

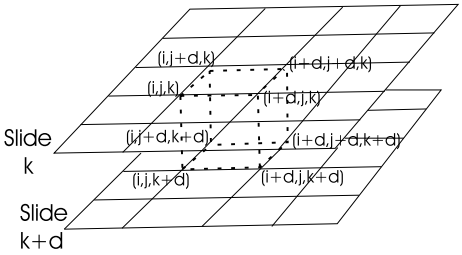


Fig. 20.2 The basic marching cubes algorithm and the order of vertices

20.2 Segmentation

Before we can proceed to model the objects, we need to segment them from the images. Segmentation techniques can be categorized into three classes [3, 4]: (a) thresholding, (b) region based, and (c) boundary based. Due to the advantages and disadvantages of each technique, many segmentation methods are based on the integration of information obtained by two techniques: the region and boundary information. Some of them embed the integration in the region detection, while others integrate the information after both processes are completed. Embedded integration can be described as integration through definition of new parameters or decision criterion for the segmentation. Postprocessing integration is performed after both techniques (boundary and region based) have been used to process the image. A different approach is the use of dynamic contours (snakes). Within each category for integration of the information, we have a great variety of methods; some of them work better in some cases, some need user initialization, some are more sensitive to noise, etc.

Due to the fact that we are dealing with medical images, we also need another important strategy: texture segmentation. Textural properties of the image can be extracted using statistical features, spatial frequency models, etc. A texture operator describes the texture in an area of the image. So, if we use a texture operator over the whole image, we obtain a new "texture feature image." In such an image, the texture of a neighborhood around each pixel is described. In most cases, a single operator does not provide enough information about the texture, and a set of operators needs to be used. This results in a set of "texture feature images" that jointly describe the texture around each pixel. The main methods for texture segmentation are Laws' texture energy filters, co-occurrence matrices, random fields, frequency-domain methods, and perceptive texture features [5, 6].

Simple segmentation techniques such as region growing, split and merge, or boundary segmentation cannot be used alone to segment tomographic images due to the complexity of the computer tomographic images of the brain. For this reason, we decide not only to combine boundary and region information (as typically is done), but to integrate information obtained from texture segmentation methods with boundary information and embed that in a region-growing strategy. A block diagram of our proposed approach is shown in Fig. 20.3.

In order to obtain the texture information, we use the Laws' texture energy masks. Laws' texture energy measures are a set of filters designed to identify specific primitive features such as spots, edges, and ripples in a local region. Laws' masks are obtained from three basic vectors, which correspond to a Gaussian, and its first and second derivatives:

$$L_3 = [1 \;\; 2 \;\; 1], \qquad E_3 = [-1 \;\; 0 \;\; 1], \qquad S_3 = [1 \;\; -2 \;\; 1]. \tag{20.1}$$

Convolution of these three vectors with themselves and with one another generates five 5×1 vectors:

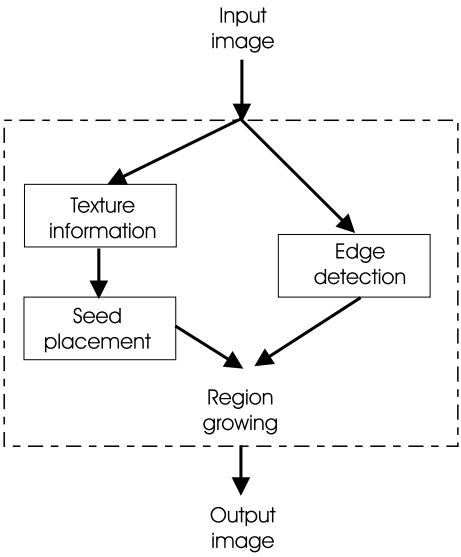


Fig. 20.3 Block diagram of our proposed approach to segment tumors in computer tomographic images

$$
\begin{aligned}
L5 &= [1\ \ 4\ \ 6\ \ 4\ \ 1]; \ E5 = [-1\ \ -2\ \ 0\ \ 2\ \ 1]; \\
S5 &= [-1\ \ 0\ \ 2\ \ 0\ \ -1]; \ W5 = [-1\ \ 2\ \ 0\ \ -2\ \ 1]; \\
R5 &= [1\ \ -4\ \ 6\ \ -4\ \ 1],
\end{aligned} \tag{20.2}
$$

which identify certain types of feature: level, edge, spot, wave, and ripple. Multiplying these five vectors by themselves and by one another produces a set of 25 unique 5×5 masks (see some examples in Fig. 20.4). Convolving these masks with an image, we obtain the so-called "texture images," which contain the response of each pixel in the image to the convoluted mask.

Our objective is to build a characteristic vector for each pixel in the image, so we take the absolute value for each pixel for each mask's result and fix the value in a position of such vector with 1 or 0, depending on whether the value is greater than zero or zero, respectively. The characteristic vector will have k values; one of them is only to identify if the pixels correspond to the background (value set to zero) or to the patient's head (value set to one), the patient's head could be skin, bone, brain, etc., while the other values are determined according to the set of masks used. If we name the characteristic vector as V_{xy} and identify each of its k coordinates as $V_{xy}[i], i = 1, \ldots, k$, the procedure could be summarized as follows:

1. For each pixel p in the CT image, if $p_{ij} \in background$, then $V[1] = 0$, else $V[1] = 1$;
2. Convolve the image with each of the texture masks to obtain a total of k "characteristic texture images";

(a)

-1	-4	-6	-4	-1
-2	-8	-12	-8	-2
0	0	0	0	0
2	8	12	8	2
1	4	6	4	1

(b)

1	-4	6	-4	1
-4	16	-24	16	-4
6	-24	36	-24	6
-4	16	-24	16	-4
1	-4	6	-4	1

(c)

-1	-4	-6	-4	-1
0	0	0	0	0
2	8	12	8	2
0	0	0	0	0
-1	-4	-6	-4	-1

(d)

-1	-2	0	2	1
0	0	0	0	0
2	4	0	-4	-2
0	0	0	0	0
-1	-2	0	2	1

Fig. 20.4 Energy texture masks used to characterize each pixel over the whole image (these masks are convolved with the image)

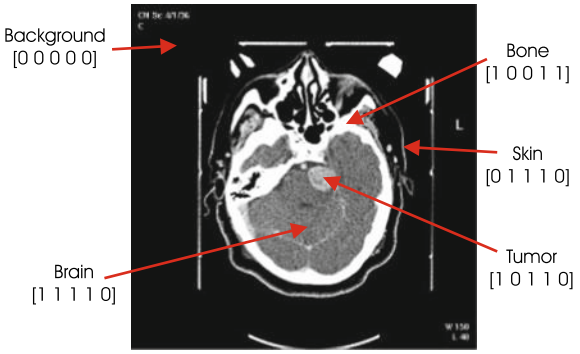


Fig. 20.5 Characteristic vectors for the main structures present in a computer tomographic image

3. For each position (x, y) in the image and for each result of the convolution of the masks, take the absolute value $absval_{xy} = \|val_{ij}\|$. If $absval_{xy} > 0$ then $V_{xy}[i] = 1$, else $V_{xy}[i] = 0$, where $i = 2, \ldots, k+1$ corresponds to the k different masks.

As a result, each structure (tissue, bone, skin, background) on the medical images has a typical vector; an example using only four masks is shown in Fig. 20.5. Obviously, the more masks used, the better the object will be characterized. It is important to note that not all the pixels belonging to the same object (i.e., the tumor) have the desired vector because of variations in values of neighboring pixels, but a good enough quantity of them do. So we can use the pixels having the characteristic vector of the object we want to extract, to establish them as seed points in a region-growing scheme.

The growing criterion (how to add neighboring pixels to the region) and the stopping criterion are as follows: compute the mean μ_{seeds} and standard deviation σ_{seeds} of the pixels fixed as seeds and then for each neighboring pixel determine whether or not it is being added to the region

$$\text{If } \sum_{1}^{k+1} (V_{xy} \oplus V_{seed}) = 0 \quad \text{or} \quad \sum_{1}^{k+1} (V_{xy} \oplus V_{seed}) = 1 \qquad (20.3)$$

$$\text{and } I(x, y) = \pm 2 * \sigma_{seeds}, \quad \text{then} \quad (x, y) \in R_t,$$

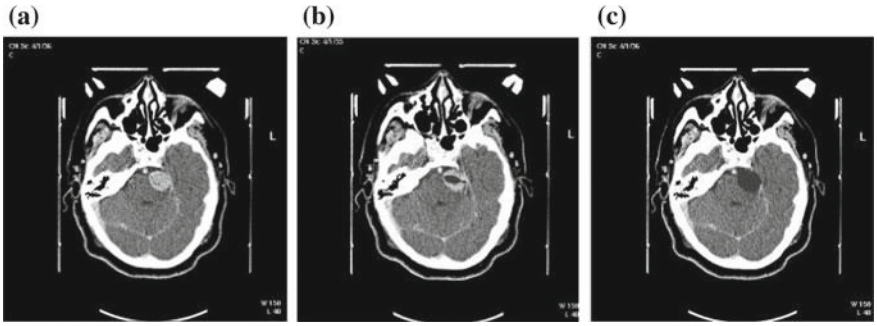


Fig. 20.6 Results for the segmentation of tumor in CT images. **a** One of the original images; **b** seed points; **c** result of segmentation

where $V_{xy} \oplus V_{seed}$ is like a XOR operator ($a \oplus b = 1$ if and only if $a \neq b$), $\sum_1^{k+1} (V_{xy} \oplus V_{seed})$ acts as a counter of different values in V_{xy} versus V_{seed} (to belong to a specific region, it must differ at most in one element), and R_t is the tumor's region. For example, suppose we use only four masks; let $V_{xy} = [0\ 1\ 0\ 1\ 1]$ and $V_{seed} = [0\ 1\ 1\ 1\ 1]$, then $V_{xy} \oplus V_{seed} = [0\ 0\ 1\ 0\ 0]$, therefore $\sum_1^{k+1} (V_{xy} \oplus V_{seed}) = 1$ and the point (x, y) is included in that region.

The region grows in all directions, but when a boundary pixel is found, the growth in that direction is stopped. Note that the boundary information helps to avoid the inclusion of pixels outside the object boundary; if we use only the texture information, wrong pixels can be added to the region, but boundary information reduces the risk of doing that.

Figure 20.6 shows an example of the process explained before. Figure 20.6a shows the original image; Fig. 20.6b shows the seed points fixed; Fig. 20.6c shows the final result, highlighting the segmented tumor in the original image.

The overall process takes only a few seconds per image and could be used to segment any of the objects, but, in our case, we focus our attention on the extraction of the tumor. After that, the next step is to model the volumetric data by some method. An example is shown in Fig. 20.7; however, in Sect. 20.1, we present a method that uses the spheres as basic entities for modeling, and in Sect. 20.3 we present our proposed approach.

20.3 Marching Spheres

Once we segment the object we are interested in, next step is to model it in 3D space. In our approach for object modeling, we will use the spheres as the basic entities, and we will follow the ideas of the marching cubes algorithm. First, we will see the 2D case. We want to approximate the shape shown in Fig. 20.1a, then the process is

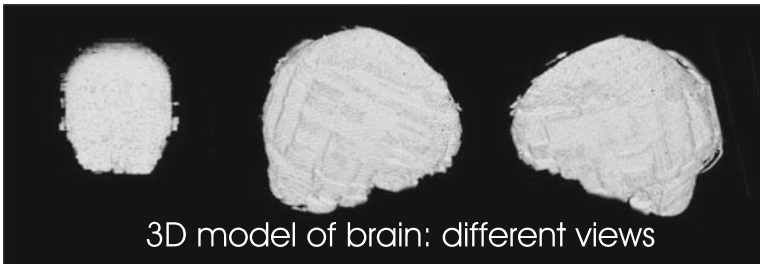


Fig. 20.7 After segmentation, 3D models are constructed. Here we show the 3D model of the brain segmented from a set of 200 MR images

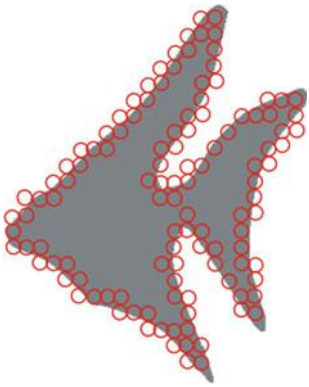


Fig. 20.8 The 2D equivalent of the modified marching cubes obtaining a representation based on circles

- Make a grid over the image and determine the points inside (or on) and outside the surface (same as in Fig. 20.1a–b).
- Draw circles centered at each inside corner connected with others out of the surface.
- Finally, draw circles centered at vertices that are connected with two vertices inside the surface.
- As a result, we obtain an approximation to the original surface (Fig. 20.8). Obviously, the finer the grid, the better the approximation.

For the 3D case, given a set of m slides, the following steps are done:

- Divide the space in logical cubes (each cube contains eight vertices, four of slide k and four of slide $k+1$).
- Determine which vertices of each cube are inside (or on) and outside the surface.
- Define the number of spheres of each cube according to Fig. 20.9 taking the indices of the cube's corners as shown in Fig. 20.2. The magnitude of the radius is $r_{p_i} = \frac{d}{2}$ for the smallest spheres $S^j_{p_i}$, $r_{m_i} = \frac{d}{2}$ for medium size spheres $S^j_{m_i}$, and $r_{g_i} = d$ for the biggest spheres $S^j_{g_i}$.

Note that we use the same 15 basic cases of the marching cubes algorithm because a total of 256 cases can be obtained from this basis. Also note that instead of triangles,

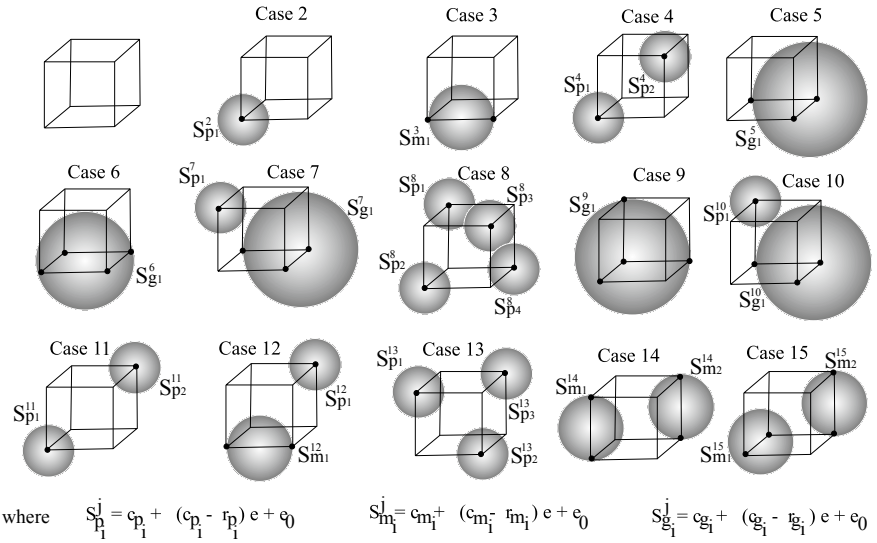


Fig. 20.9 The 15 basic cases of surface intersecting cubes. The cases are numbered in this figure starting with the upper left corner, from 1 to 15

we define spheres and that our goal is not to have a good render algorithm, but to have a representation of the volumetric data based on spheres, which, as we said before, could be very useful in the process of object registration.

20.3.1 Experimental Results for Modeling

To test the algorithm, we used several images such as 26 images of a human skull containing balloons to simulate the brain, 36 images of a real patient, 16 images of a real tumor in a patient, and other different CT images. The first step is to segment the brain (tumor) from the rest of structures. Once the brain (tumor) is properly segmented, the images are binarized to make the process of determining if each pixel is inside or outside the surface easier. The Canny edge detector is used to obtain the boundary points of each slide; by this way, we can compare the number of such points (which we call n) with the number of spheres obtained, also giving a comparison with the number of primitives obtained with the Union of Spheres algorithm.

Figure 20.10 shows one CT image of the skull with balloons, the segmented object, and the approximation of the surface by circles using the 2D version of our approach. Figures 20.11 and 20.12 show the CT of a real patient and tumor segmented. Figure 20.13a shows the results for the 3D case modeling the brain of our human head model (balloons) extracted from CT images, while Fig. 20.13b shows

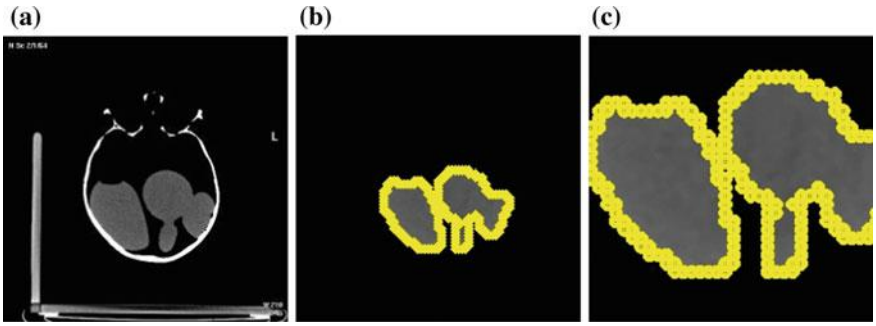


Fig. 20.10 Simulated brain: **a** original of one CT slide; **b** segmented object and its approximation by circles according to the steps described in Sect. 20.3; **c** zoom of (**b**)

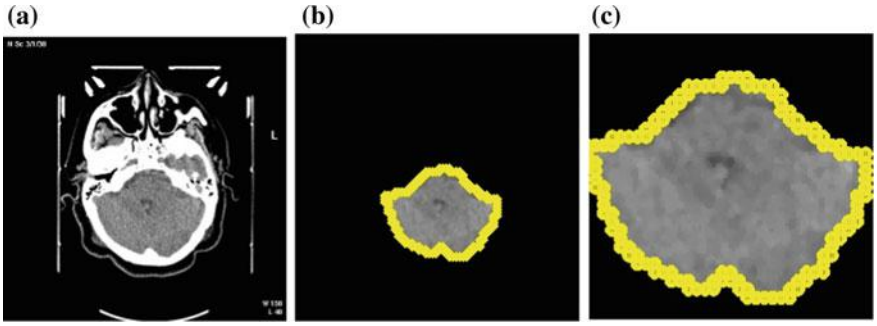


Fig. 20.11 Real patient: **a** original of one CT slide; **b** brain segmented and its approximation by circles; **c** zoom of (**b**)

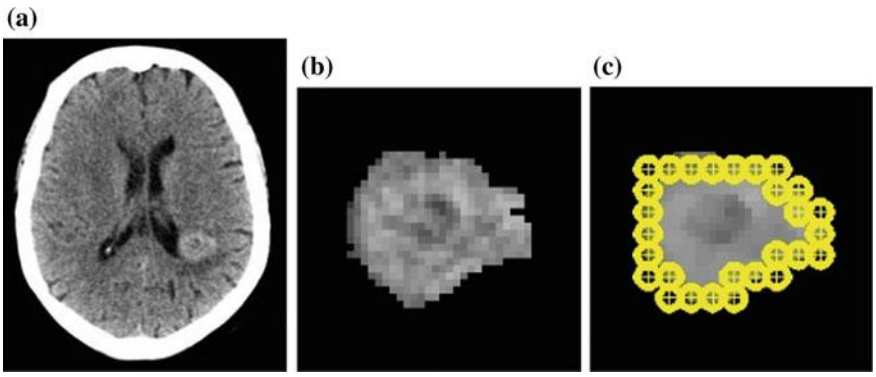


Fig. 20.12 Real patient: **a** original of one CT slide; **b** zoom of the tumor segmented; **c** tumor approximation by circles

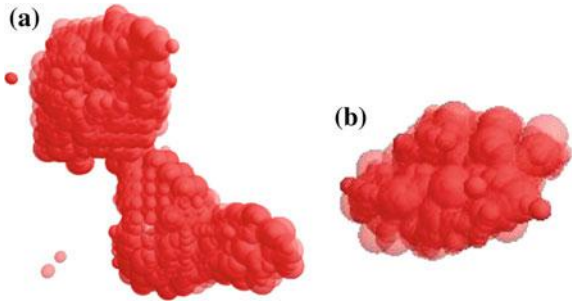


Fig. 20.13 Approximation of shape of three-dimensional objects with marching spheres; **a** approximation of the brain structure extracted from CT images (synthetic data); **b** approximation of the tumor extracted from real patient data

Table 20.1 Comparison between the number of spheres using the approach based on Delaunay tetrahedralization and our approach based on the marching cubes algorithm (called marching spheres); n is the number of boundary points; d is the distance between vertices in logical cubes of our approach

n/d	No. of spheres in DT approach	No. of spheres in marching spheres
3370/1	13,480	11,866
3370/3	8642	2602
10329/2	25,072	8362
2641/1	6412	5008

the 3D model of the tumor from the real patient (extracted from 16 CT images). Table 20.1 is a comparison between the results of the Union of Spheres and our approach for the case of brain being modeled. The first row shows the worst case with both approaches and the second row shows the number of spheres with improvements in both algorithms (reduction of spheres in DT is done by grouping spheres in a single one that contains the others, while such a reduction is done using a displacement of $d = 3$ in our approach). The number of boundary points was $n = 3370$ in both cases. The reduction in the number of primitives obtained with our approach, while maintaining clear enough the representation, results evident.

20.4 Registration of Two Models

In [7], a comparison between a version of the popular Iterated Closest Point (ICP) [8] and the Thin-Plate Spline-Robust Point Matching (TPS-RPM) algorithms for non-rigid registration appears. TPS-RPM performed better because it can avoid getting trapped in local minima, and it aligns the models even in the presence of a large

amount of outliers. However, the authors had used only sets of 2D and 3D points. Now we have spheres modeling the object, and these spheres have not only different centers, but also different radii. This fact prompted us to look at the representation of spheres using the Conformal Geometric Algebra (CGA), which is a geometric algebra of five dimensions. In next section, we explain how to register two models based on spheres using their representation in CGA.

20.4.1 Sphere Matching

The registration problem appears frequently in computer vision and medical image processing. Suppose we have two point sets and one of them results from the transformation of the other, but we don't know the transformation nor the correspondences between the points. In such a situation, we need an algorithm that finds these two unknowns as well as possible. If, in addition, the transformation is nonrigid, the complexity increases enormously. In the variety of registration algorithms existing today, we find some that assume knowledge of one of these unknowns and solve for the other one, but there are two that solve for both: Iterated Closest Point (ICP) and Thin-Plate Spline-Robust Point Matching (TPS-RPM). It has been shown [9] that TPS-RPM gives better results than ICP; therefore, we will adapt such an algorithm for the alignment of models based on spheres. It is important to note that previous works only deal with 2D and 3D points, but now we have spheres in the 5D space of the CGA.

The notation is as follows: $\mathbf{V} = \{S_j^I\}$, $j = 1, 2, \ldots, k$ is the set of spheres of the first model, or model at time t_1; $\mathbf{X} = \{S_i^F\}$, $i = 1, 2, \ldots, n$ is the set of spheres of the second model, or model at time t_2 (expected model); $\mathbf{U} = \{S_i^E\}$, $i = 1, 2, \ldots, n$ is the set of spheres resulting after the algorithm finishes (estimated set). The super index denotes if the sphere S belongs to the initial set (I), the expected set (F), or the estimated set (E).

To solve the correspondence problem means finding the correspondence matrix Z, (Eq. (20.4)), which indicates which sphere of the set $\mathbf{V}$ is transformed into which sphere of the set $\mathbf{X}$ by some transformation f. Thus, if we correctly estimate the transformation f and apply it to X ($U = f(V)$), then we should obtain the set X. To deal with outliers, the matrix Z is extended as in (20.4). The inner part of Z (with size $k \times n$) defines the correspondences.

$$
Z = \begin{array}{c|cccccc|c}
z_{ji} & S_1^F & S_2^F & S_3^F & S_4^F & \cdots & S_n^F & outlier_{n+1}^F \\
\hline
S_1^I & 1 & 0 & 0 & 0 & \cdots & 0 & 0 \\
S_2^I & 0 & 1 & 0 & 0 & \cdots & 0 & \cdots \\
S_3^I & 0 & 0 & 1 & 0 & \cdots & 0 & \cdots \\
\cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
S_k^I & 0 & 0 & 0 & 0 & \cdots & 1 & 0 \\
outlier_{k+1}^I & 0 & 0 & 0 & 1 & \cdots & 0 &
\end{array} \tag{20.4}
$$

However, to deal with outliers using binary correspondence matrices can be very cumbersome [7]. For this reason, TPS-RPM uses two techniques to solve the correspondence problem:

Soft assign: The basic idea is to allow the matrix of correspondences M to take continuous values in the interval [0, 1]; this "fuzzy" correspondence improves gradually during optimization without jump in the space of permutations of binary matrices.

Deterministic annealing: Technique used to control the fuzzy correspondences by means of an entropy term in the cost function (called energy function), introducing a parameter T of temperature that is reduced in each stage of the optimization process beginning at a value T_0 and reducing by a factor r until some final temperature is reached.

Using these two techniques, the problem is to minimize the function:

$$\begin{aligned} E(M, f) = &\sum_{i=1}^{n}\sum_{j=1}^{k} m_{ji} \left|S_i^F - f(S_j^I)\right|^2 + \lambda |Lf|^2 \\ &+ T\sum_{i=1}^{n}\sum_{j=1}^{k} m_{ji}\log m_{ji} - \varsigma \sum_{i=1}^{n}\sum_{j=1}^{k} m_{ji}, \end{aligned} \tag{20.5}$$

where m_{ji} satisfy the condition $\Sigma_{i=1}^{n+1} m_{ji} = 1$ for $j \in \{1, 2, \ldots, k+1\}$, $\Sigma_{j=1}^{k+1} m_{ji} = 1$ for $i \in \{1, 2, \ldots, n+1\}$, and $m_{ji} \in [0, 1]$. The parameter λ is reduced in an annealing scheme $\lambda_i = \lambda_{init} T$. The basic idea in this heuristic is that more global and rigid transformations should be first favored with large values of λ. We will follow the next two steps:

Update the correspondences: For the spheres S_j^I, $j = 1, 2, \ldots, k$, and S_i^F, $i = 1, 2, \ldots, n$, modify m_{ji}:

$$m_{ji} = \frac{1}{T} e^{-\frac{(S_i^F - f(S_j^I))^2}{T}}, \tag{20.6}$$

for outliers $j = k+1$ e $i = 1, 2, \ldots, n$:

$$m_{k+1,i} = \frac{1}{T_0} e^{-\frac{(S_i^F - f(S_{k+1}^I))^2}{T_0}}, \tag{20.7}$$

and for outliers $j = 1, 2, \ldots, k$ e $i = n+1$:

$$m_{j,n+1} = \frac{1}{T_0} e^{-\frac{(S_{n+1}^F - f(S_j^I))^2}{T_0}}. \tag{20.8}$$

Update transformation: To update the centers, we use

$$\begin{aligned} E_{tps}(D, W) = &\|Y_c - V_c D - \Phi_c W\|^2 + \lambda_1 (W^T \Phi_c W) \\ &+ \lambda_2 [D - I]^T [D - I], \end{aligned} \tag{20.9}$$

where V_c is the matrix given by $V_c = \{(S_j^I \wedge E) \cdot E\}$, $j = 1, 2, \ldots, k$, Y_c is interpreted as the new estimated position of the spheres, Φ_c is constructed by $\phi_b = (S_b^I \wedge E) \cdot E - (S_a^I \wedge E) \cdot E$; therefore, Φ_c contains information about the structure of the sets; D, W represent the rigid and nonrigid deformations, respectively, affecting the centers of the spheres, and are obtained by solving

$$f((S_a^I \wedge E) \cdot E | D, W) = ((S_a^I \wedge E) \cdot E)D + \phi((S_a^I \wedge E) \cdot E)W. \tag{20.10}$$

The radii are updated using a dilator, which is defined as

$$D_\lambda = e^{\frac{-\log(\lambda) \wedge E}{2}} = e^{-\frac{\log(\rho_{j_{\max}}/\rho_{i_{\max}}) \wedge E}{2}}, \tag{20.11}$$

where $\rho_{j_{\max}}$ and $\rho_{i_{\max}}$ are the radii of the more corresponding pair of spheres, according to matrix M. That is, the index i of the initial set and j of the expected set where the i-th row of M has a maximum value $\max(m_{ij})$. Dilators are applied as

$$S'^I_j = D_\lambda S_j^I D_\lambda. \tag{20.12}$$

The annealing process helps to control the dual update process. Parameter T is initialized in T_{init} and is gradually reduced as $T_i = T_{init} r$, where r is the annealing rate until some final temperature is reached. Parameters λ_1 and λ_2 follow an annealing scheme as mentioned earlier.

20.4.2 Experimental Results for Registration

Figures 20.14 and 20.15 show two examples registering sets of spheres. Figures 20.14a and 20.15a are the initial set or representation at time t_1; Figs. 20.14b and 20.15b are the deformed or expected set, or representation at time t_2. These two representations should be registered. Figure 20.14c shows the results of the registration process. Note that researchers usually use TPS-RPM with 2D or 3D vectors because they cannot go beyond such a dimension; in contrast, using conformal geometric algebra we have a homogeneous representation that preserves isometries and uses the sphere as the basic entity. Note that the algorithm adjusted the radius, as expected (this is not possible using only 3D vectors).

We carried out other experiments. Table 20.2 shows the average errors between the corresponding spheres' centers, measured with the expected and resulting sets. This error is measured as

$$\epsilon = \frac{\sum_{i,j} \sqrt{c(i,j)(S_i^F - S_i^E)^2}}{N}, \tag{20.13}$$

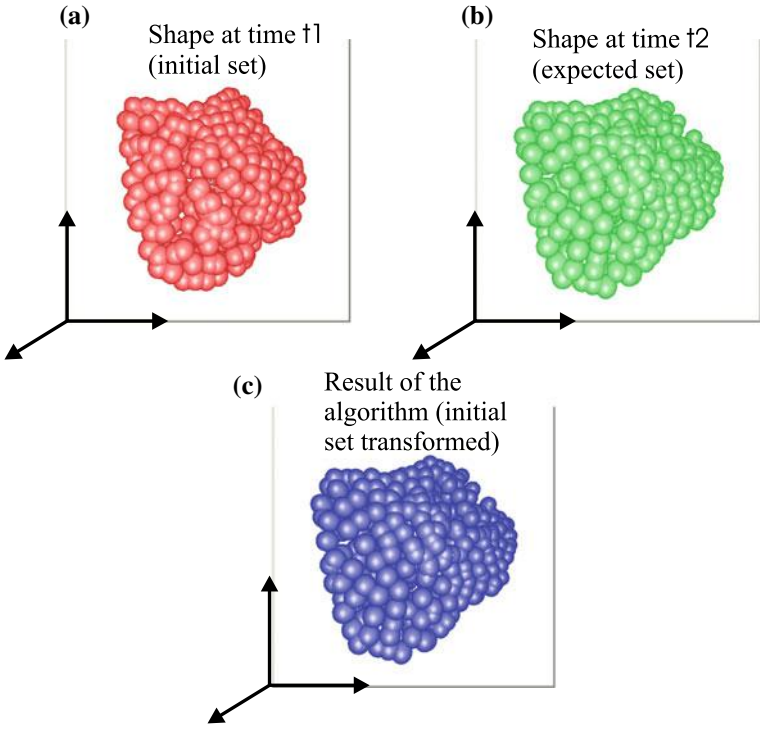


Fig. 20.14 Registration of models based on spheres: **a** initial set (or representation at time t_1); **b** expected set (or representation at time t_2); **c** result of the deformation of the initial model to match the second one

Table 20.2 Average errors measured as the distance (in voxels) between the centers of corresponding spheres, calculated with the expected and resulting sets, and according to (20.13)

Number of spheres of the initial set	Number of spheres of the expected set	Average error according to (20.13)
60	60	12.86
98	98	0.72
500	500	2.80
159	143	18.04
309	280	15.63
340	340	4.65

where N is the number of pairs of corresponding spheres; $c(i, j) = 1$ if S_i^F corresponds to S_i^E or $c(i, j) = 0$ if they do not correspond.

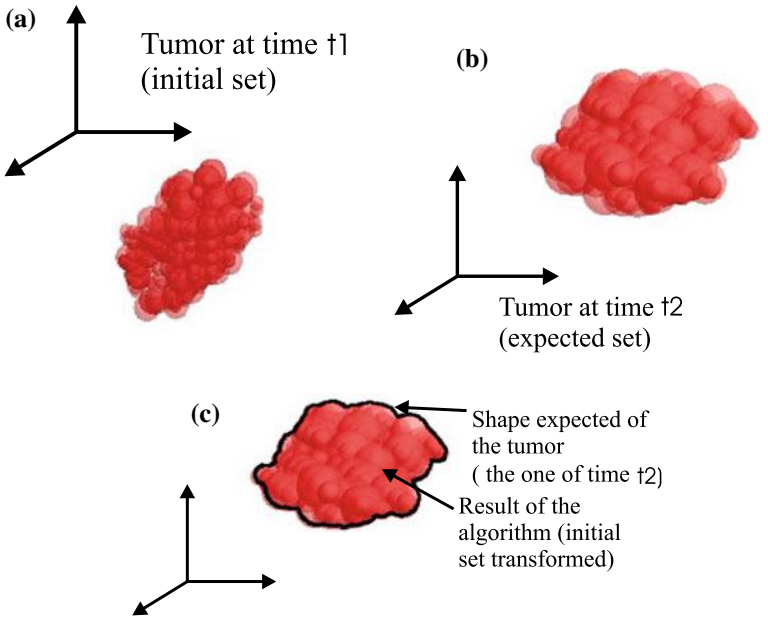


Fig. 20.15 Registration of models based on spheres for a tumor: **a** initial shape of the tumor; **b** expected shape of the tumor (representation at time t_2); **c** result after registration (transformed initial model)

20.5 Conclusions

The chapter first presents an approach for medical image segmentation that combines texture and boundary information and embeds it into a region-growing scheme, having the advantage of integrating all the information in a simple process. We also show how to obtain a representation of volumetric data using spheres. Our approach, called *marching spheres*, is based on the ideas exposed in the marching cubes algorithm but it is not intended for rendering purposes or displaying in real time, but rather for reducing the number of primitives modeling the volumetric data. With this saving, it makes a better registration process; better in the sense of using fewer primitives, which reduces the registration errors. Also, the chapter shows how to represent these primitives as spheres in the conformal geometric algebra, which are five-dimensional vectors that can be naturally used with the principles of TPS-RPM. Experimental results seem to be promising.

References

1. Ranjan, V., & Fournier, A. (1995). Union of spheres (UoS) model for volumetric data. In *Proceedings of the 11th Annual Symposium on Computational Geometry*, Vancouver, Canada, C2–C3 (pp. 402–403).
2. Lorensen, W., & Cline, H. (1987). Marching cubes: A high resolution 3D surface construction algorithm. *Computer Graphics*, *21*(4), 163–169.
3. Muñoz, X. (2002). *Image segmentation integrating color, texture and boundary information.* Ph.D. thesis, University of Girona, Girona, Spain.
4. Fu, K. S., & Mui, J. K. (1980). A survey on image segmentation. *Pattern Recognition*, *12*, 395–403.
5. Chantler, M. J. (1994). *The effect of variation in illuminant direction on texture classification*. Ph.D. thesis, Department of Computing and Electrical Engineering, Heriot-Watt University.
6. Tamura, H., Mori, S., & Yamawaki, T. (1980). Textural features corresponding to visual perception. *IEEE Transactions on Systems, Man and Cybernetics*, *8*(6), 460–473.
7. Chui, H., & Rangarajan, A. (2000). A new point matching algorithm for non-rigid registration. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (Vol. 2, pp. 44–51).
8. Zhang, Z. (1992). Iterative point matching for registration of free-form curves. Technical report 1658, INRIA.
9. Bayro-Corrochano, E., & Rivera-Rovelo, J. (2004). Non-rigid registration and geometric approach for tracking in neurosurgery. In *International Conference on Pattern Recognition*, Cambridge, UK (pp. 717–720).

Chapter 21
Geometric Computing for Minimal Invasive Surgery

In this chapter, we show the treatment of a variety of tasks of medical robotics handled using a powerful, non-redundant coefficient geometric language. This chapter is based on our previous works [1, 2]. You will see how we can treat the representation and modeling using geometric primitives like points, lines, and spheres. The screw and motors are used for interpolation, grasping, holding, object manipulation and surgical maneuvering. We use geometric algebra algorithms in three scenarios: the virtual world for surgical planning, the haptic interface to command the robot arms, and the visually guided robot arms system for operation of ultrasound scanning and surgery. Note that in this work, we do not present a complete system for computer-aided surgery, here we illustrate the application of geometric algebra algorithms for some relevant tasks in minimal invasive surgery.

Figure 21.1 shows the scenario using a phantom of kidney, robot arms, a tracking system, haptic interface, and computers for monitoring and control. The operative system for the software is done using the popular Robotic Operative System (ROS) in Linux. Next subsections present in detail the different tasks tackled using real-time algorithms designed in the conformal geometric algebra. You can appreciate the elegance, geometric intuitiveness, and power of this kind of modern geometric computing which is run in state-of-the-art equipment. Currently, we are accelerating our algorithms using C++ and Gaalop [3] with FPGA or Nvidia cards. Figure 21.1 shows the simulated kinematics of the used Motoman robot arm and the Barret hand [4] for the kidney surgery system.

Due to the length limitation of the paper, we explain only some tasks of the vision robot system, with the objective to highlight the performance of the most interesting algorithms using conformal geometric algebra.

© Springer Nature Switzerland AG 2020

E. Bayro-Corrochano, *Geometric Algebra Applications Vol. II*,
https://doi.org/10.1007/978-3-030-34978-3_21

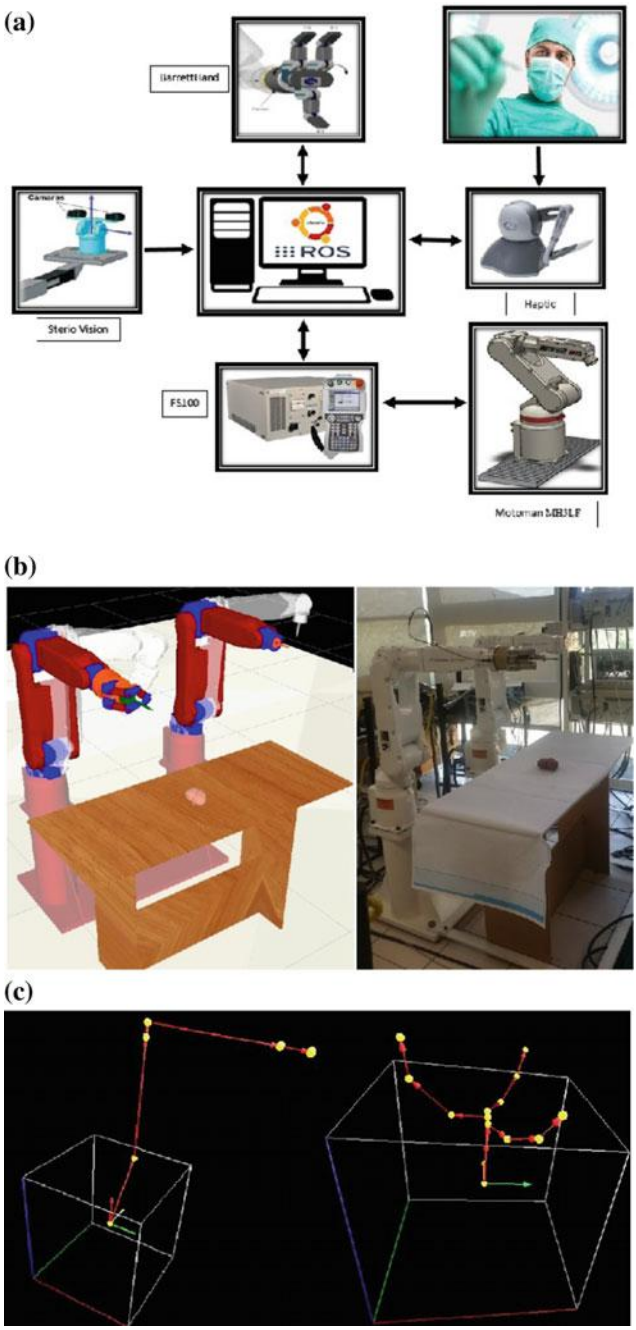


Fig. 21.1 **a** The medical robotic system. **b** The robot arm system and kidney phantom (virtual and real scenarios). **c** Kinematics of the Motoman robot arm and Barret hand

21.1 Tracker Endoscope Calibration

The surgery scenario of Fig. 21.2 shows that the endoscope and Ultrasound Probe US need to be tracked. Next, we explain the procedure for the hand–eye calibration which is needed to track the coordinate system of the endoscope. In Fig. 21.2, it is shown that there is a (rigid) transformation between the calibration grid and the Polaris system, $\boldsymbol{M}_{B_g}$. Such transformation will be used to validate the results of the endoscope-tracking system calibration method. The transformations involved in the problem are expressed as motors of the CGA: $\boldsymbol{M} = \boldsymbol{T}\boldsymbol{R}$. They and the coordinate systems involved are as follows:

- $\mathbf{O_p}$: Reference frame of the Polaris system.
- $\mathbf{O_{m_e}}$: Reference frame of the spherical markers.
- $\mathbf{O_{c_e}}$: Reference frame of the endoscopic camera.
- $\mathbf{O_{g_c}}$: Reference frame of the calibration grid.
- $\boldsymbol{M}_{A_i}$: Transformation from $\mathbf{O_{g_c}}$ to $\mathbf{O_{c_e}}$ in the i-th movement of the camera. This transformation is given by the extrinsic parameters of the camera.
- $\boldsymbol{M}_{B_i}$: Transformation from $\mathbf{O_{m_e}}$ to $\mathbf{O_p}$. It is obtained by the optical tracking system.
- $\boldsymbol{M}_X$: Transformation from $\mathbf{O_{c_e}}$ to $\mathbf{O_{m_e}}$, obtained with the endoscope-tracking calibration algorithm.
- $\boldsymbol{M}_a\boldsymbol{B}_g$: Transformation from $\mathbf{O_{g_c}}$ to $\mathbf{O_p}$. It is obtained directly from the Polaris system by placing spherical markers on the plane where the calibration grid lies on.

We compute the motor $\boldsymbol{M}$ using the motor algebra SVD method for hand–eye calibration [5].

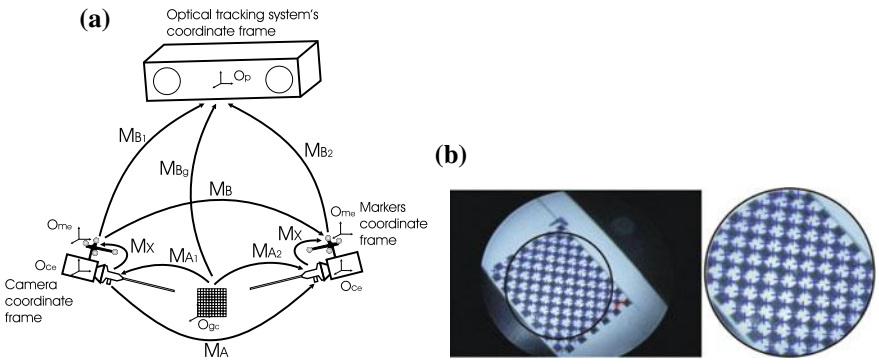


Fig. 21.2 **a** The problem of the calibration between the endoscopic camera and the optical tracking system (camera infrared Polaris). **b** Result after the hand–eye calibration of the grid projection onto the camera image

21.2 Planning Surgical Paths in Virtual World

During a preoperative stage, the surgeons plan possible paths for the surgical instruments. Figure 21.4 shows the trajectory of a curve using the interpolation on Study's quadric method of Sect. 9.1. Note that between the segments, the transitions are smooth, as opposite to the effect using the linear interpolation MSLERP of Eq. (9.5) and shown in Fig. 21.3d, where we can identify certain peaks distorting the smoothness of the path.

These paths will be followed autonomously by the robot arms. We are currently developing robust algorithms using fuzzy nonlinear control techniques, so that the robot arms will follow the trajectories accurately and safely. Also an hand two haptic interfaces the surgeon can help to the robot arms to modify the maneuvering. This will be reported in next article.

Figure 21.3a–d shows in the virtual world the digitalized form of a real human kidney. The surgeon planes in the virtual world on the kidney surface the pose paths to be followed for the US probe and for the tumor resection. For this, the surgeon selects

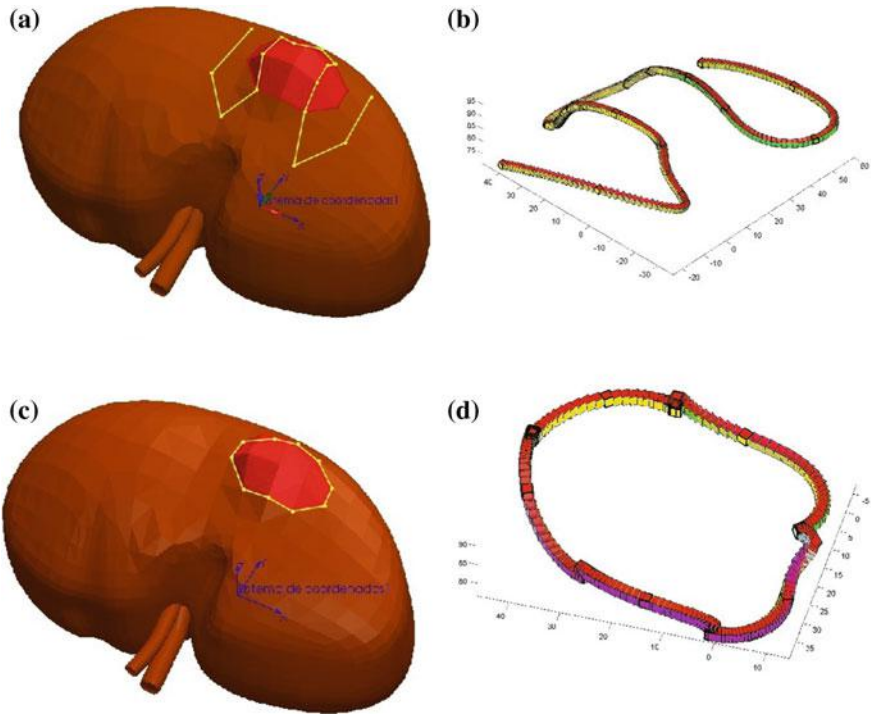


Fig. 21.3 Trajectories on a kidney human model. **a** Chosen points for the US trajectory, **b** interpolated points for the path of the 3D pose of the US probe, **c** chosen points for the tumor resection, **d** MSLERP-interpolated points for the path of the 3D pose of the surgical tool for the resection of the tumor

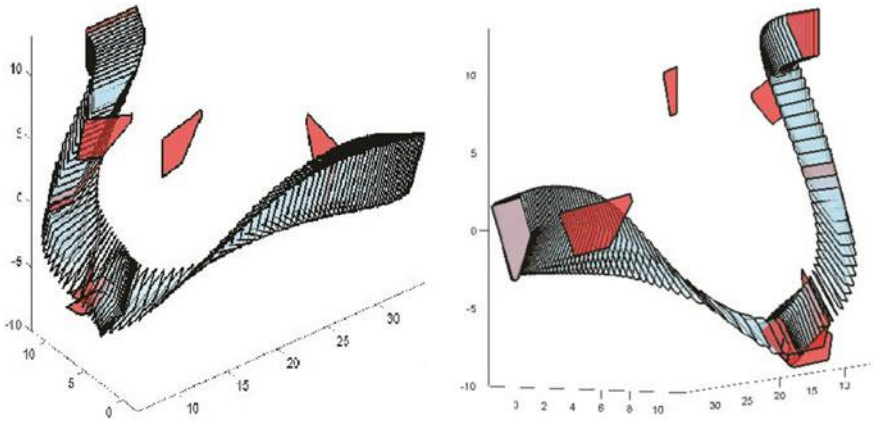


Fig. 21.4 Interpolation of the surgery knife pose along a smooth path using Studys quadric

certain points normal to the kidney surface for possible trajectories of the pose of the instruments. Using these set of points, we utilize the MSLERP algorithm of Eq. (9.5) to find a smooth pose path for both the US probe and for the resection of a tumor. Figure 21.3b–d shows interpolated trajectories for the US probe and for the tumor resection. Such trajectories can be improved using Study's quadric interpolation method, with which it is possible to obtain smoother paths on trajectory.

21.3 Execution of Surgical Maneuvers

In order to illustrate our approach, we carried out some essential tasks needed in the kidney surgery. For the experiment, we use a kidney phantom. It is the 3D plastic model printed using our digitized model of a real human kidney, see Fig. 21.5a.

During the intra-operative phase, the surgeons can monitor in the virtual world the execution of US inspection and tumor resection in the real world. As we said previously, this can be done more efficient and safe by including the haptic devices in the control loop. Note that, we use two stereo vision system, one is to track the endoscope which carries the US probe and surgical instruments and also a RGB-D device like a Kinect which senses the evolution of the surgical operation and measures position and velocity of the instruments which are fed back to close the control loop. This is a matter of current work. What we present now here is the execution of the tumor resection in real time along a planned path done in the virtual world. This trajectory is passed to the robot arm which follows this trajectory in real time. For sake of simplification, we are using an improvised surgical knife which is fastened in a cube support hold by the Barret hand which grasps the knife. In Fig. 21.5b–f, we show in virtual and real worlds, a sequence of the trajectory during the resection of a tumor.

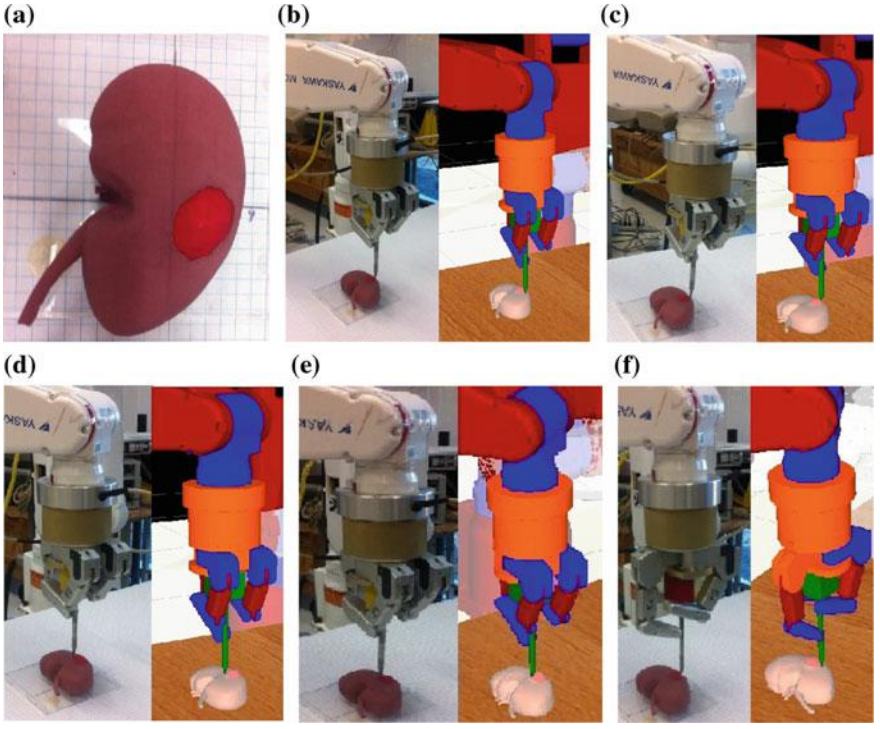


Fig. 21.5 **a** Tumor in a kidney phantom. **b** Instances during the real-time resection of the tumor shown in real and virtual world

The selected essential procedures to be performed during a surgery of a kidney are the following:

– Registration of the organ,
– Measurement of elasticity in a region,
– Procedure for a suture,
– Handling the motion of the ultrasound probe, and
– Suture of a wound.

The procedures are implemented using the geometry of the task in question, so that thanks to our geometric language we can represent the tasks in such manner that the intuition of the surgeon will not be disturbed rather our human–machine interface will enhance the experience and intuition of the surgeon, for that geometric algebra is an ideal language for such purpose. In fact, as knowledge engineers, we describe the task in words and use figures drawn by hand, then we derive the equations in geometric algebra using the symbolic language Maple18 together with the package for geometric algebra called eClifford [6] to derive the procedure, and then this algorithm is rewritten in C++ and its speed has been increased using FPGA or CUDA

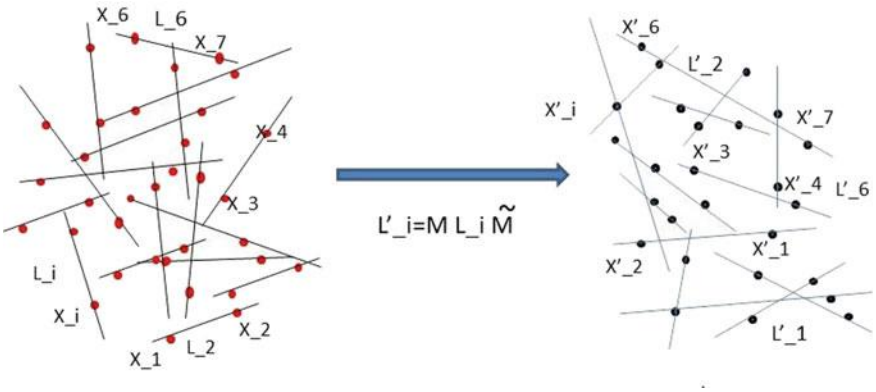


Fig. 21.6 Registration of lines using the hand–eye calibration method

Nvidia. Our work on speeding up our geometric algorithms was submitted [7]. In this paper, we do not include the Maple18 code due to limited paper length. We include as illustration in one task just a short version of the Maple18 code.

21.4 Registration of the Organ

The patient kidney after X-radiography and tomography is represented in a virtual world. For a surgery, one needs to register the pose of the real kidney with its representation in the virtual world. During the surgery, the patient kidney may move and it is needed to register these motions with the pose of the virtual kidney. After the registration is done, the surgeon can plane key cuts and surgery paths. A practical way to register the poses robust against noise is the use of line sets in the real and virtual worlds and compute the SE(3) rigid transformation or motor $\boldsymbol{M}$ between the sets of lines (Fig. 21.6)

$$\boldsymbol{L}_i' = \boldsymbol{M}\boldsymbol{L}_i\tilde{\boldsymbol{M}}, \tag{21.1}$$

where $\boldsymbol{L}_i$ and $\boldsymbol{L}_i$ are the $i = 1, \ldots, N$ lines on the real and the virtual kidney. To compute $\boldsymbol{M}$, we utilize the hand–eye calibration method explained in Sect. 16.1.2. The lines are computed taken couple of points in correspondence. Having computed the motor $\boldsymbol{M}$, we can correct the pose of the virtual kidney in accordance with the real one. Any point on the virtual point can be readjusted as follows:

$$X_i = \mathbf{M}\, X_i\, \widetilde{\mathbf{M}}. \tag{21.2}$$

In the ROS environment, we use the $\boldsymbol{M}$ to synchronize the pose of the real and virtual kidneys. If the patient moves, one can recompute the pose changes in real

time simply using new sets of lines which are in correspondence. The couple of points is taken by a sensor applied directly to the kidney and since the motion of the robot manipulator is known, one can establish the point correspondence of the pair, the points or lines, and then compute on and on the motor $\boldsymbol{M}$ using the hand–eye calibration method.

21.5 Representing the Elasticity on the Organ Surface

It is important to know the elasticity of the kidney for the surgery example around a tumor or a suspicious tissue. Furthermore, the elasticity helps to recreate the virtual model of the kidney which will regulate the force of the haptic interaction. For this goal, the surgeon using a sensor measures in certain points p_i the elasticity k_i of the surface of the organ sampled uniformly either along a path or on a 2D surface.

In each Euclidean point p_i, a sphere is located with a ratio equal to the elasticity coefficient k_i, i.e.,

$$S_i = p_i + \left(\frac{c_i^2 - k_i^2}{2}\right) e_\infty + e_0. \tag{21.3}$$

After the sampled points are associated with spheres, an interpolation of spheres is carried out using the explained algorithm *Motor Iterative Blending* 1 of Sect. 9.2. Thus, one obtains a set of spheres which interpolate the elasticity of the organ tissue along a path, see Fig. 21.7.

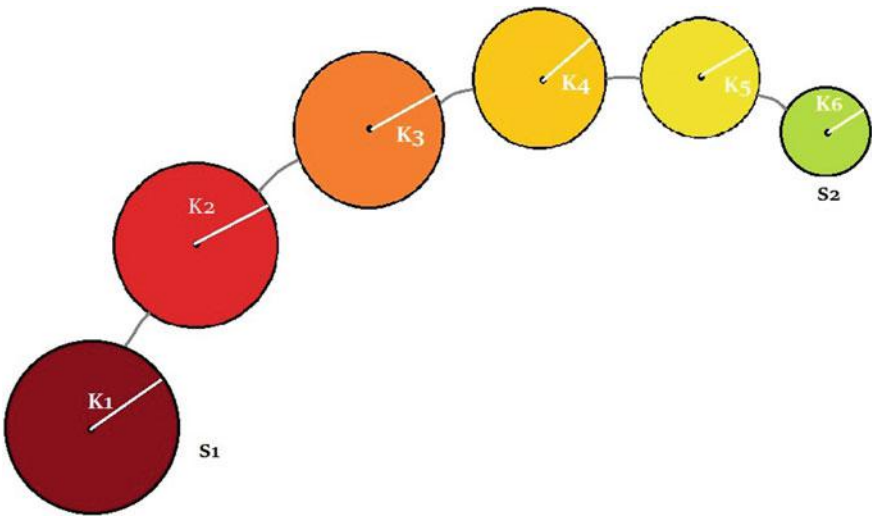


Fig. 21.7 Sphere interpolation along a 1D path with changing ratios which represent the elasticity coefficient k_i

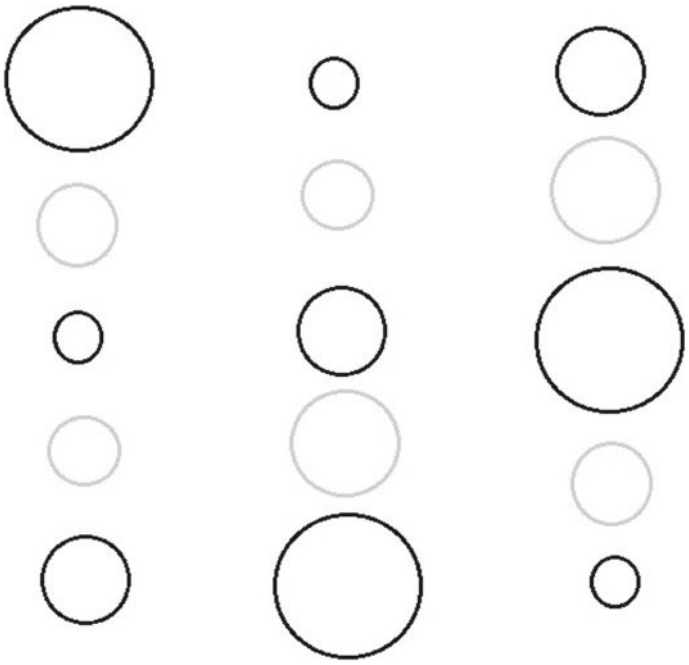


Fig. 21.8 Sphere interpolation along a 2D surface

If the sampled points are distributed homogeneously on a 2D surface, the effect is even better as shown in Fig. 21.8.

21.6 Procedure for Surgery Cuts

For the surgery cut, it is important to know the pose of the instrument with respect to the organ surface, see Fig. 21.9.

To compute the incidence between the scalpel and the organ surface, one takes the plane of the surface represented by three points $\phi = p_i \wedge p_j \wedge p_k$ and for the scalpel incidence line two points $L = p_1 \wedge p_2$ which describe its contour as shown in Fig. 21.10.

The equation representing this relation reads

$$\tilde{\pi} L + \tilde{L} \pi. \tag{21.4}$$

If the equation equals zero, it means that the scalpel is on the kidney surface and if it is < 0, the scalpel has penetrated the organ.

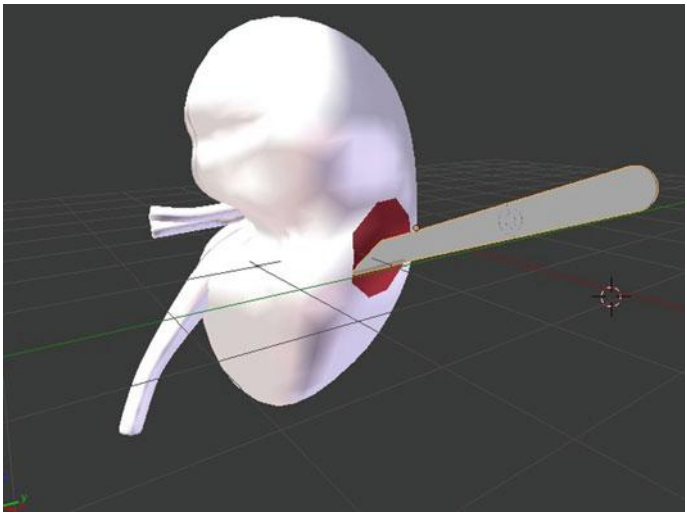


Fig. 21.9 Representation of the scalpel, the kidney in the virtual world

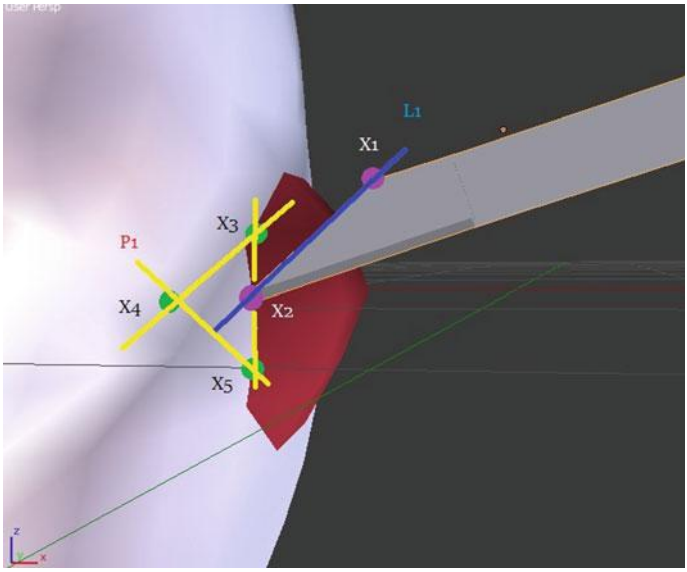


Fig. 21.10 Set the points to guide the incidence of the cut

Equation (21.4) is very intuitive and represents the geometric relation between a line and a plane of the motor algebra $G^{+}_{3,0,1}$. However, for the implementation, we utilize an affine plane where points are $x^h = x_e + e_o$. In a simulation, we computed this using Maple 18 and eClifford [6]. First one computes the direct distance d between the line $L^h = a^h_1 \wedge a^h_2$ and the plane $\pi^h = b^h_1 \wedge b^h_2 \wedge b^h_3$

$$
\begin{aligned}
&d[a^h_1 \wedge a^h_2, b^h_1 \wedge b^h_2 \wedge b^h_3] \equiv \\
&\{\overline{e} \cdot (a^h_1 \wedge a^h_2)\} \wedge \{\overline{e} \cdot (b^h_1 \wedge b_2 \wedge b^h_3)\}]^{-1} \\
&[\overline{e} \cdot (a^h_1 \wedge a^h_2 \wedge b^h_1 \wedge b^h_2 \wedge b^h_3)] \\
&= [(a_2 - a_1) \wedge (b_2 - b_1) \wedge (b_3 - b_2)]^{-1} \\
&[(a_2 - a_1) \wedge (b_1 - a_2) \wedge (b_2 - b_1) \wedge (b_3 - b_2)].
\end{aligned}
\tag{21.5}
$$

Note that the incidence relation between a line $L^h = a^h_1 \wedge a^h_2$ and the plane $\pi^h = b^h_1 \wedge b^h_2 \wedge b^h_3$ is also determined by its join, $L^h \cup \pi^h$. Clearly, if the join is a trivector, the line L^h lies on the plane π^h. The other possibility is that the join is the pseudoscalar $I = \sigma_{123} e$, thus

$$
p = L^h \cap \pi^h = L^h \cdot (\pi^h)^* = L^h \cdot (\pi^h \cdot \overline{I}). \tag{21.6}
$$

Now, if $d = 0$, the line L^h intersects the plane in the point p Eq. 21.6 and if $d \neq 0$, it means that the line is parallel to the plane π, see Fig. 21.11. If one knows a reference point X_1 on the line, with the point of the intersection P, one computes the distance $h = \|P - X_1\|$ from contour of the US prove organ.

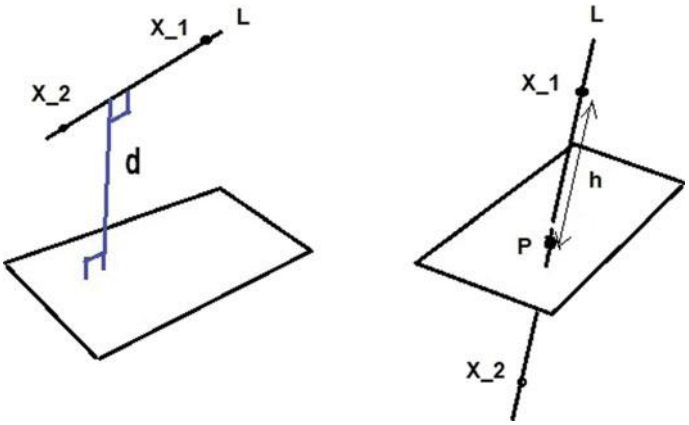


Fig. 21.11 The direct distance between a line and a plane. P is the intersection point of line and a plane and h is the distance between P and the reference point X_1

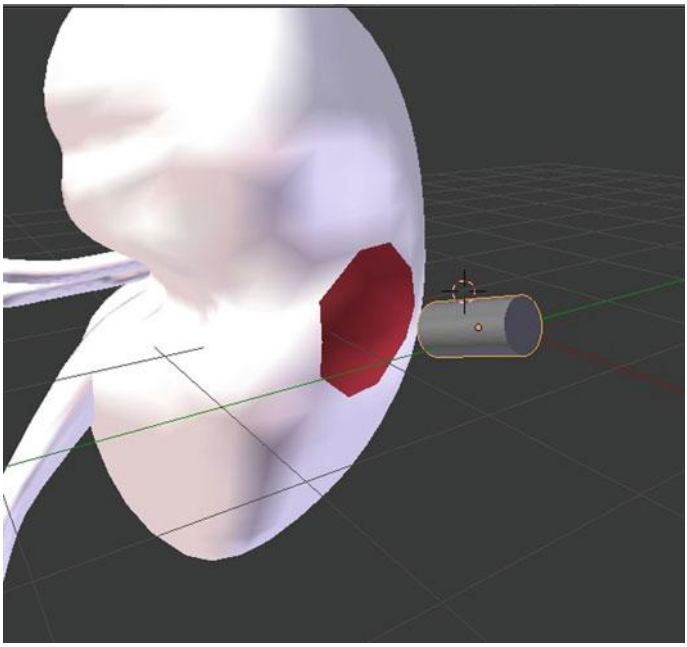


Fig. 21.12 Kidney and US probe in the virtual world

21.6.1 Handling the US Probe

The surgery is supported by inspection of the tissue using a Ultrasound Probe (USP), see Fig. 21.12. One must know precisely the pose of the US to update the virtual world to support the haptic interaction, see Fig. 21.13. In order to compute the pose, one can exploit the geometric relation between two planes. One plane computes with three points $Xus_{1,2,3}$ on the face of the US probe, which gives

$$\pi_1 = Xus_1 \wedge Xus_2 \wedge Xus_3, \tag{21.7}$$

and a plane on the organ surface computes with other three points as well, which gives

$$\pi_2 = Xkn_1 \wedge Xkn_2 \wedge Xkn_3. \tag{21.8}$$

Figure 21.14 shows the planes on the US probe and the organ. The contact is computed as follows:

$$\pi_1\pi_2 + \pi_2\pi_1, \tag{21.9}$$

which is zero when the planes are in contact.

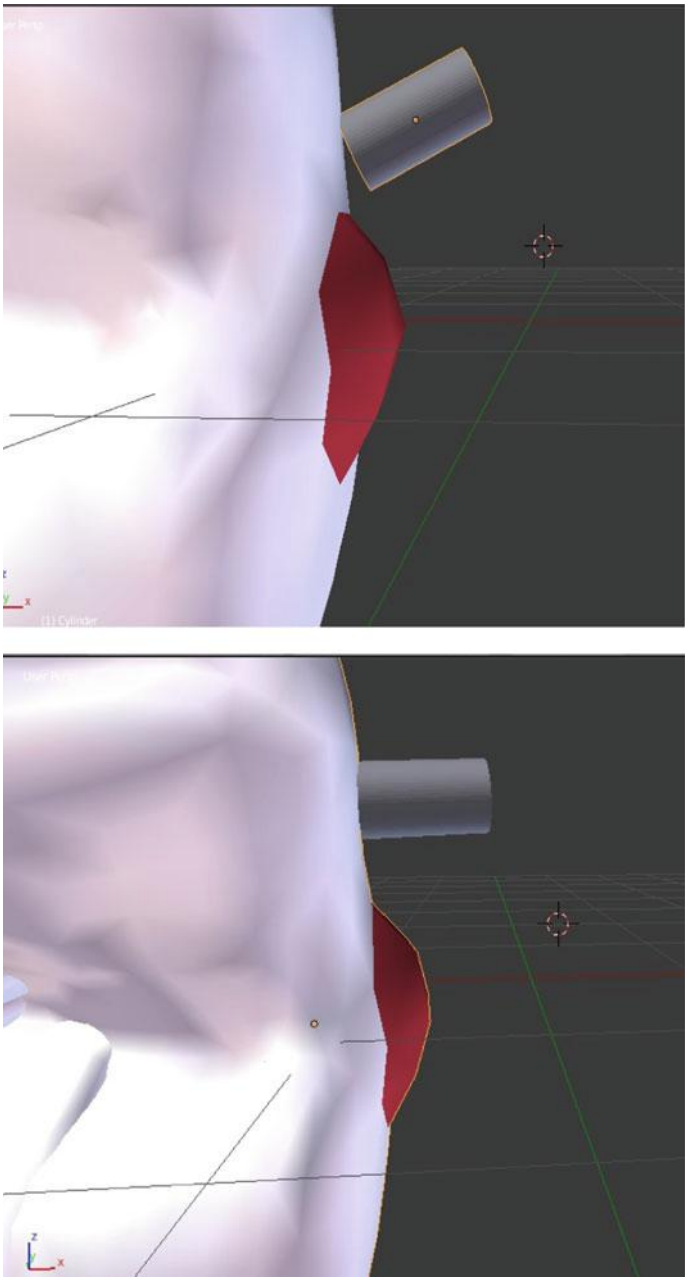


Fig. 21.13 Poses of contact between the US probe and the organ

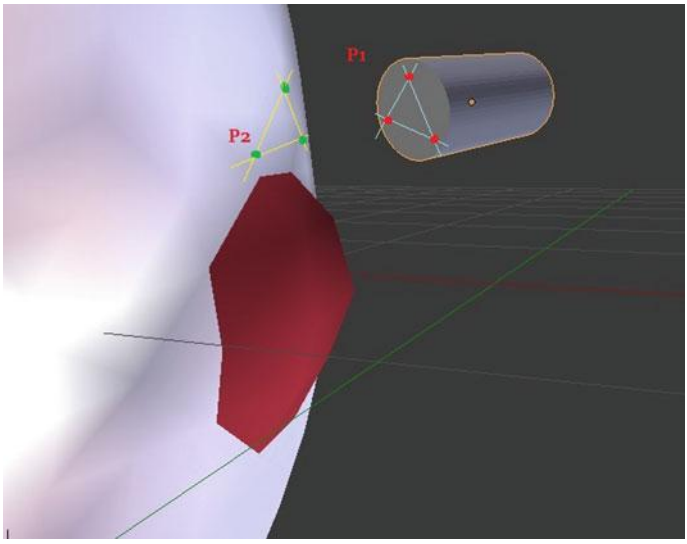


Fig. 21.14 Representation of the incidence relations between planes

Equation (21.9) is very intuitive and it represents the geometric relation between plane π_1 and plane π_2 using motor algebra $G^+_{3,0,1}$. Similarly as Sect. 21.6, one takes a couple of points on the plane π_1 to compute a line L^h_1 and three points of the other plane π^h_2, and hence using this line one can measure whether this line of the plane is parallel to the plane π_2 or if this line intersects the plane. For the implementation, we used the affine plane $x^h = x_e + e_o$, Maple 18 y eClifford [6]. Firstly, one computes the direct distance d between the line $L^h_1 = a^h_1 \wedge a^h_2$ and the plane $\pi^h_2 = b^h_1 \wedge b^h_2 \wedge b^h_3$

$$
\begin{aligned}
&d[a^h_1 \wedge a^h_2, b^h_1 \wedge b^h_2 \wedge b^h_3] \equiv \\
&\{\overline{e} \cdot (a^h_1 \wedge a^h_2)\} \wedge \{\overline{e} \cdot (b^h_1 \wedge b_2 \wedge b^h_3)\}]^{-1} \\
&[\overline{e} \cdot (a^h_1 \wedge a^h_2 \wedge b^h_1 \wedge b^h_2 \wedge b^h_3)] \\
&= [(a_2 - a_1) \wedge (b_2 - b_1) \wedge (b_3 - b_2)]^{-1} \\
&[(a_2 - a_1) \wedge (b_1 - a_2) \wedge (b_2 - b_1) \wedge (b_3 - b_2)].
\end{aligned}
\tag{21.10}
$$

Note that the incidence relation between the line $L^h_1 = a^h_1 \wedge a^h_2$ and the plane $\pi^h_2 = b^h_1 \wedge b^h_2 \wedge b^h_3$ is also determined by the join, $L^h \cup \pi^h$. Clearly, if the join is a trivector, the line L^h lies on the plane π^h. The other possibility is that its join e is the pseudoscalar $I = \sigma_{123} e$, in this case

$$
p = L^h_1 \cap \pi^h_2 = L^h_1 \cdot (\pi^h_2)^* = L^h_1 \cdot (\pi^h_2 \cdot \overline{I}). \tag{21.11}
$$

Note that one used the line L^h_1 of the first plane π^h_1, so depending upon if $d = 0$ the line L^h_1 lies on the plane π^h_1 or intersects the plane ϕ^h_2 in the line L according to

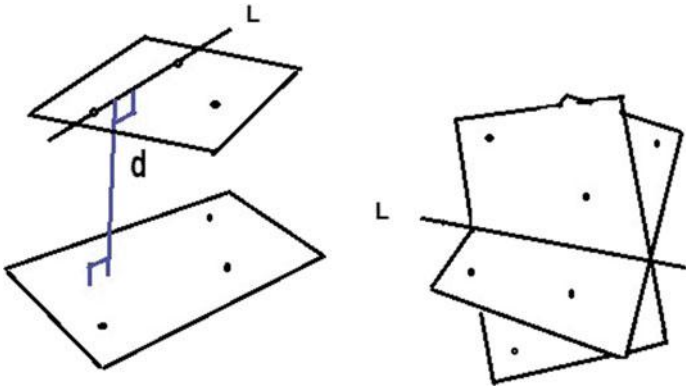


Fig. 21.15 (left) The direct distance between a plane π_1^h and other plane π_1^2. (right) The intersection line between the two planes

Eq. (21.11). If $d \neq 0$ means that the plane ϕ_1^h where the line lies is parallel to the plane π_2^h, see Fig. 21.15.

21.6.2 Suture

The last step after a surgery is to suture the wound. For that some points are assigned around the wound. These points are laser pointed and transferred to the virtual organ. Figure 21.16 shows some selected points. Figure 21.17 exemplifies how the suture goes following sequentially opposite points.

First, we have to compute a translator $\boldsymbol{T}$ to transport a point to another, as follows:

$$T = 1 + e_\infty \left(\frac{X_i - X_j}{2} \right), \tag{21.12}$$

where X_i y X_j are a couple of points to bring close, and then we carry out the translation of the needle to the opposite point, namely,

$$Z_i = T Z_i \tilde{T}. \tag{21.13}$$

Figure 21.18 shows the final step of the surgery when the wound has been closed.

Equation (21.13) represents the simple step to translate a point to a goal on an opposite point. Yet, if we consider that during the suture the needle has to pull the string, for that we compute a rotor $\boldsymbol{R}$ to move the needle around a semicircle. Figure 21.19a shows the geometry of this operation, where an hand of three points a tangent plane is computed. Using the bisector, one gets the center of rotation, so that the needle moves along a semicircle to the opposite point. Figure 21.19b shows

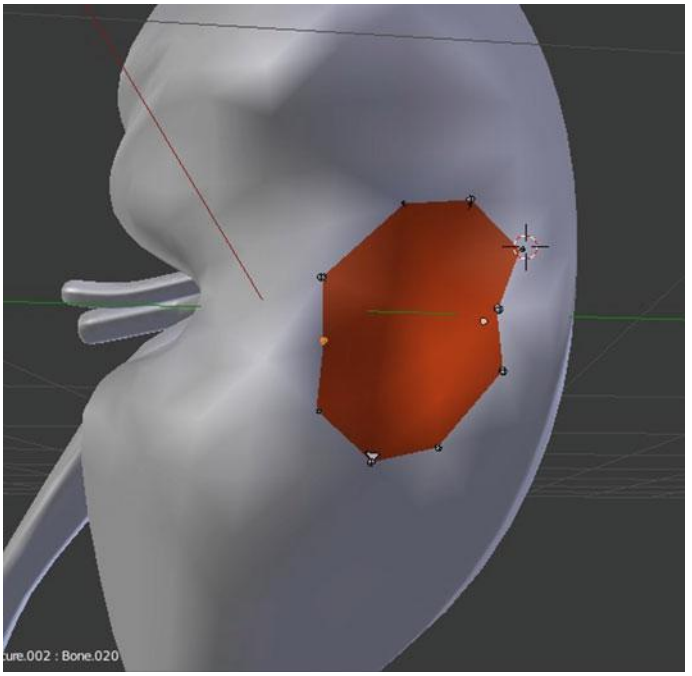


Fig. 21.16 Selected points shown in the virtual kidney

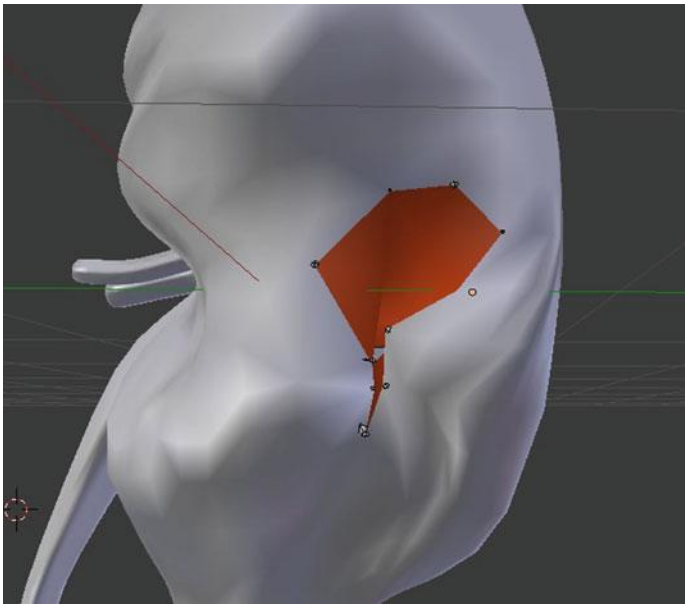


Fig. 21.17 Suture step by step of the kidney wound

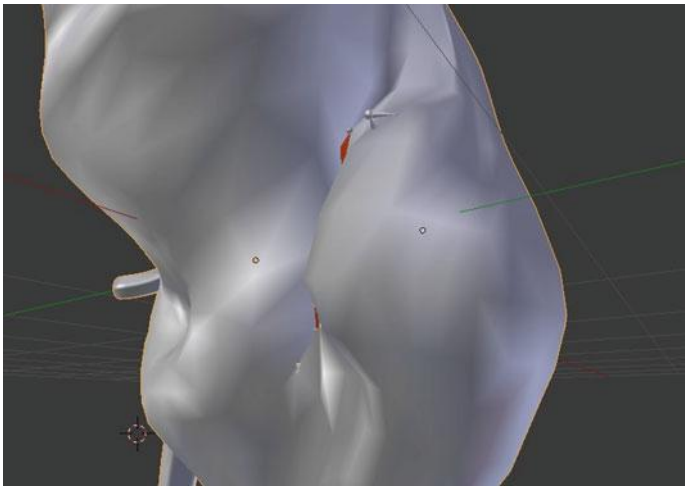


Fig. 21.18 Kidney after the suture

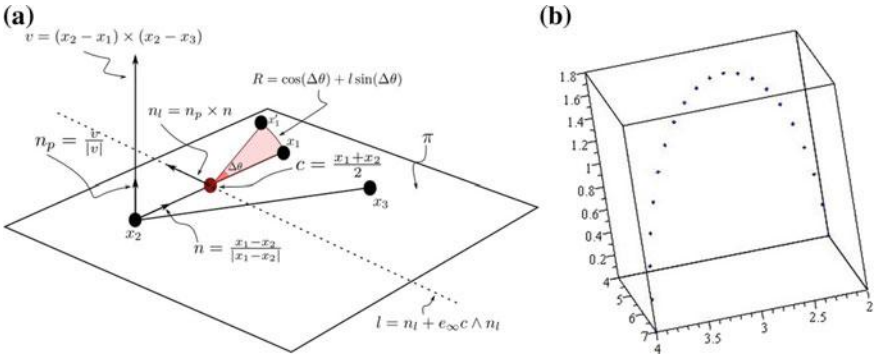


Fig. 21.19 **a** Computing the rotor $\boldsymbol{R}$. **b** Semicircle of the trajectory which is followed by the needle

the needle trajectory during a single step of the suture. As illustration, we include the Maple code developed for the suture. Next the procedure for the case of needle motion for wound surgery, use ten points.

```
CloseWound := proc (x::eclimon, eclibasmon, eclipolynom)
Define Variables
N := 10; M := 10; (ten points)
X := Matrix(3, 10, [[1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
[10, 9, 8, 7, 6, 5, 4, 3, 2, 1], [2, 4, 5, 1, 2, 3, 4, 7, 9, 2]]);
Xn := Matrix(3, 10, [[10, 9, 8, 7, 6, 5, 4, 3, 2, 1],
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10], [2, 4, 5, 1, 2, 3, 4, 7, 9, 2]]);
for i from 1 by 1 to N-2
do p1 := X[1, i]*e[1] + X[2, i]*e[2] + X[3, i]*e[3];
```

```
p2 := X[1, i+1]*e[1] + X[2, i+1]*e[2] + X[3, i+1]*e[3];
p3 := X[1, i+2]*e[1] + X[2, i+2]*e[2] + X[3, i+2]*e[3];
pm := 0.5*(p1+p2);
n := (p1–p2);
n := n/sqrt(einner(n, n));
nx := coeff(n, e[1]); ny := coeff(n, e[2]); nz := coeff(n, e[3]);
v := crossvec(p2–p1, p3–p1);
np := v/sqrt(einner(v, v));
nl := crossvec(np, n);
for j from 1 by 1 to M
do
theta := (j/M)*(pi/2);
(R, Rc) := rotorL(theta: nx, ny, nz, nl);
pn := ecmul(R, p1, Rc);
Xn[1, j] := coeff(np, e[1]);
Xn[2, j] := coeff(np, e[2]);
Xn[3, j] := coeff(np, e[3]);
end do;
end do;
end do;
return ecmul(ewedge(x, E), E)
end proc;
```

For the application in real time, this code has to be translated in C++ or Python language.

21.7 Conclusions

This work introduces a modern and nonconventional geometric language to support and to enhance the handling of different tasks in medical robot vision. The authors present the use of geometric algebra techniques to handle hand–eye calibration, 3D model registration, interpolation, virtual reality and graphics engineering, intuitive haptic, maneuvering, grasping, and object manipulation. These tasks are essential for vision-based medical robotics. Our case of study is the automation of kidney surgery. The main contribution of this work is to show the development of geometric algorithms using geometric algebra as a vehicle between the surgeon, haptics, and the organ in the virtual and real world. This language helps the surgeon's approach by stimulating the surgeon's intuition utilizing the geometric entities and geometric properties of the organ and the surgery itself.

This work uses conformal geometric algebra to solve some key computational issues in medical robot vision without the need to leave the mathematical framework. We present the development of geometric algorithms for a real scenario of non-invasive kidney surgery, where different tasks of the perception, control, and maneuvering are formulated and solved in conformal geometric algebra. Note that

we do not report a complete surgery, just we illustrate the applications of geometric algorithms for some important tasks in a non-invasive kidney surgery. This verifies how useful can be the use of conformal geometric algebra for the development of algorithms for this purpose. The experimental analysis shows promising possibilities for the use of this powerful geometric language to handle a variety of tasks in medical robotics. This powerful geometric language can be used to handle multiple tasks in minimal invasive medical robotics. Moreover, this language can be used to develop algorithms to assist the training of surgeons. Readers can use this geometric language for different applications in graphic engineering and robotics as well.

References

1. Ureña-Ponce, O., Soria-García, G., Martínez-Terán, G., Ortega-Cisneros, S., & Bayro-Corrochano, E. (2019). Accelerating interpolation in conformal geometric algebra using GPU. *Journal of Advances on Applications of Clifford Algebra*.
2. Bayro-Corrochano, E., Garza-Burgos, A. M., & Del Valle-Padilla, J. L. (2019). Geometric intuitive techniques for human machine interaction in medical robotics. *Journal of Social Robotics*.
3. Hildenbrand, D., Pitt, J., & Koch, A. (2010). Gaalop–high performance parallel computing based on conformal geometric algebra. In E. Bayro-Corrochano, G. Sheuermann (Eds.), *Geometric algebra computing for engineering and computer science* (Chap. 22, pp. 477–494). Springer-Verlag.
4. Barrett-Technology. http://www.barrett.com/robot/index.htm.
5. Bayro-Corrochano, E. (2010). *Geometric computing for wavelet transforms, robot vision, learning, control and action*. London: Springer-Verlag.
6. Ablamowicks, R. eCLIFFORD Software packet using Maple for Clifford algebra. Computations. http://math.tntech.edu/rafal.
7. Soria-Garcia, G., Garza-Burgos, M., Ureña-Ponce, O., Ortega-Cisneros, S., & Bayro-Corrochano, E. (2017). Speed up of conformal geometric entities interpolation using FPGA. *Advances in Applied Clifford Algebras*.

Appendix A
Notation

This section includes the list of the notation with a brief description.

$\mathbb{R}$	The real numbers
$\mathbb{S}^n$	The unit sphere in $\mathbb{R}^{n+1}$
$\mathbb{C}$	The complex numbers
$\mathbb{H}$	The quaternion algebra
$\mathbb{R}^n$	A vector space of dimension n over the field $\mathbb{R}$ with Euclidean signature
$\mathbb{R}^{p,q}$	A vector space of dimension $n = p + q$ over the field $\mathbb{R}$ with signature (p, q)
$\mathbb{R}^{m\times n}$	The direct product $\mathbb{R}^m \otimes \mathbb{R}^n$
$\mathcal{G}_n$	The geometric algebra over $\mathbb{R}^n$
$\mathcal{G}_{p,q}$	The geometric algebra over $\mathbb{R}^{p,q}$
$\mathcal{G}_{p,q,r}$	The special or degenerated geometric algebra over $\mathbb{R}^{p,q}$
$\mathcal{G}^k_{p,q}$	The k-vector space of $\mathcal{G}_{p,q}$
x	Scalar element
$\mathbf{x}$	Vector of $\mathbb{R}^n$
$\boldsymbol{x}$, $\boldsymbol{X}$	Vector and multivector of geometric algebra
$\boldsymbol{X}_k$	A blade of grade k
$\mathcal{F}$	Flag, a set of geometric entities of an object frame

© Springer Nature Switzerland AG 2020

E. Bayro-Corrochano, *Geometric Algebra Applications Vol. II*,
https://doi.org/10.1007/978-3-030-34978-3

Operator symbols

$\boldsymbol{X}\boldsymbol{Y}$	Geometric product of the multivectors $\boldsymbol{X}$ and $\boldsymbol{Y}$
$\boldsymbol{X} * \boldsymbol{Y}$	Scalar product of $\boldsymbol{X}$ and $\boldsymbol{Y}$
$\boldsymbol{X} \cdot \boldsymbol{Y}$	Inner product of $\boldsymbol{X}$ and $\boldsymbol{Y}$
$\boldsymbol{X} \wedge \boldsymbol{Y}$	Wedge product of $\boldsymbol{X}$ and $\boldsymbol{Y}$
$\boldsymbol{X} \bigvee \boldsymbol{Y}$	Meet operation of $\boldsymbol{X}$ and $\boldsymbol{Y}$
$\boldsymbol{X} \bigwedge \boldsymbol{Y}$	Join product of $\boldsymbol{X}$ and $\boldsymbol{Y}$
$\boldsymbol{X}^{-1}$	Inverse of $\boldsymbol{X}$
$\langle \boldsymbol{X} \rangle_k$	Projection of $\boldsymbol{X}$ onto grade k
$\boldsymbol{X}^*$	Dual of $\boldsymbol{X}$
$\widetilde{\boldsymbol{X}}$	Reverse of $\boldsymbol{X}$
$\boldsymbol{X}^\dagger$	Conjugate of $\boldsymbol{X}$
$\|\boldsymbol{X}\|$	Norm of $\boldsymbol{X}$
$P^{\parallel}_{\langle \boldsymbol{Y} \rangle_l}(\langle \boldsymbol{X} \rangle_k)$	Projection of $\langle \boldsymbol{X} \rangle_k$ onto $\langle \boldsymbol{Y} \rangle_l$
$P^{\perp}_{\langle \boldsymbol{Y} \rangle_l}(\langle \boldsymbol{X} \rangle_k)$	Projection of $\langle \boldsymbol{X} \rangle_k$ onto $\langle \boldsymbol{Y} \rangle_l$

Appendix B
Useful Formulas for Geometric Algebra

Geometric Product

$\boldsymbol{x}\boldsymbol{y} = \boldsymbol{x} \cdot \boldsymbol{y} + \boldsymbol{x} \wedge \boldsymbol{y}$
$\boldsymbol{y}\boldsymbol{x} = \boldsymbol{x} \cdot \boldsymbol{y} - \boldsymbol{x} \wedge \boldsymbol{y}$
$\boldsymbol{x} \cdot \boldsymbol{X}_r = \frac{1}{2}(\boldsymbol{x}\boldsymbol{X}_r - (-1)^r \boldsymbol{X}_r \boldsymbol{x})$
$\boldsymbol{x} \wedge \boldsymbol{X}_r = \frac{1}{2}(\boldsymbol{x}\boldsymbol{X}_r + (-1)^r \boldsymbol{X}_r \boldsymbol{x})$

Multivectors

$\boldsymbol{X} = \langle \boldsymbol{X} \rangle_0 + \langle \boldsymbol{X} \rangle_1 + \cdots = \sum_r \langle \boldsymbol{X} \rangle_r$
$\boldsymbol{X}_r \boldsymbol{Y}_s = \langle \boldsymbol{X}_r \boldsymbol{Y}_s \rangle_{|r-s|} + \langle \boldsymbol{X}_r \boldsymbol{Y}_s \rangle_{|r-s|+2} + \cdots + \langle \boldsymbol{X}_r \boldsymbol{Y}_s \rangle_{r+s}$
$\boldsymbol{X}_r \cdot \boldsymbol{Y}_s = \langle \boldsymbol{X}_r \boldsymbol{Y}_s \rangle_{|r-s|}$
$\boldsymbol{X}_r \cdot \lambda = 0$ for scalar λ
$\boldsymbol{X}_r \wedge \boldsymbol{Y}_s = \langle \boldsymbol{X}_r \boldsymbol{Y}_s \rangle_{r+s}$
$\langle \boldsymbol{X}\boldsymbol{Y} \rangle = \langle \boldsymbol{Y}\boldsymbol{X} \rangle$
$\boldsymbol{X}_r \cdot (\boldsymbol{Y}_s \cdot \boldsymbol{Z}_t) = (\boldsymbol{X}_r \wedge \boldsymbol{Y}_s) \cdot \boldsymbol{Z}_t \; r + s \leq t$ and $r, s > 0$
$\boldsymbol{X}_r \cdot (\boldsymbol{Y}_s \cdot \boldsymbol{Z}_t) = (\boldsymbol{X}_r \cdot \boldsymbol{Y}_s) \cdot \boldsymbol{Z}_t \; r + t \leq s$
$\boldsymbol{X} \times (\boldsymbol{Y} \times \boldsymbol{Z}) + \boldsymbol{Z} \times (\boldsymbol{X} \times \boldsymbol{Y}) + \boldsymbol{Y} \times (\boldsymbol{Z} \times \boldsymbol{X}) = 0$

Products of frequent use of vectors $\boldsymbol{x}$, $\boldsymbol{y}$, z, $\boldsymbol{w}$ and a bivector X

$\boldsymbol{x} \cdot (\boldsymbol{y} \wedge \boldsymbol{x}) = \boldsymbol{x} \cdot \boldsymbol{y}z - \boldsymbol{x} \cdot z\boldsymbol{y}$
$(\boldsymbol{x} \wedge \boldsymbol{y}) \cdot (z \wedge \boldsymbol{w}) = \boldsymbol{x} \cdot \boldsymbol{w}\boldsymbol{y} \cdot z - \boldsymbol{x} \cdot z\boldsymbol{y} \cdot \boldsymbol{w}$
$\boldsymbol{x} \cdot (\boldsymbol{Y} \cdot \boldsymbol{X}) = (\boldsymbol{x} \wedge \boldsymbol{y}) \cdot \boldsymbol{X}$
$(\boldsymbol{x} \wedge \boldsymbol{y}) \times (z \wedge \boldsymbol{w}) = \boldsymbol{y} \cdot z\boldsymbol{x} \wedge \boldsymbol{w} - \boldsymbol{x} \cdot z\boldsymbol{y} \wedge \boldsymbol{w} + \boldsymbol{x} \cdot \boldsymbol{w}\boldsymbol{y} \wedge z - \boldsymbol{y} \cdot \boldsymbol{w}\boldsymbol{x} \wedge z$
$(\boldsymbol{x} \wedge \boldsymbol{y}) \times \boldsymbol{X} = (\boldsymbol{x} \cdot \boldsymbol{X}) \wedge y + \boldsymbol{x} \wedge (\boldsymbol{y} \cdot \boldsymbol{X})$

© Springer Nature Switzerland AG 2020

E. Bayro-Corrochano, *Geometric Algebra Applications Vol. II*,
https://doi.org/10.1007/978-3-030-34978-3

Bivectors

$$\boldsymbol{XY} = \boldsymbol{X}\cdot\boldsymbol{Y} + \boldsymbol{X}\times\boldsymbol{Y} + \boldsymbol{X}\wedge\boldsymbol{Y}$$
$$\boldsymbol{YX} = \boldsymbol{X}\cdot\boldsymbol{Y} - \boldsymbol{X}\times\boldsymbol{Y} + \boldsymbol{X}\wedge\boldsymbol{Y}$$
$$\boldsymbol{Y}\times\boldsymbol{X}_r = \langle \boldsymbol{Y}\times\boldsymbol{X}_r\rangle_r$$
$$\boldsymbol{Z}\times(\boldsymbol{XY}) = \boldsymbol{Z}\times\boldsymbol{XY} + \boldsymbol{XZ}\times\boldsymbol{Y}, \quad \forall\, \boldsymbol{X},\boldsymbol{Y}$$

Pseudoscalar I

$$\boldsymbol{X}_r\cdot(\boldsymbol{Y}_s I) = \boldsymbol{X}_r \bigwedge \boldsymbol{Y}_s I \quad r+s\le n$$
$$\boldsymbol{X}_r \bigwedge(\boldsymbol{Y}_s I) = \boldsymbol{X}_r\cdot\boldsymbol{Y}_s I \quad r\le s$$

Reflections, Rotations

$$\boldsymbol{X}_r \to (-1)^r n\boldsymbol{X}_r n$$
$$\boldsymbol{X}_r \to \boldsymbol{R}\boldsymbol{X}_r\widetilde{\boldsymbol{R}}$$
$$\boldsymbol{R} = e^{-\boldsymbol{B}/2} = kl\cdots mn$$
$$\boldsymbol{R}\widetilde{\boldsymbol{R}} = \widetilde{\boldsymbol{R}}\boldsymbol{R} = 1$$

Motor Algebra: Rotors, Translators, and Motors

$$\boldsymbol{T} = 1 + I\tfrac{1}{2}\boldsymbol{t} = e^{-\frac{\boldsymbol{t}}{2}I}$$
$$\boldsymbol{R} = \cos\left(\tfrac{\theta}{2}\right) + \sin\left(\tfrac{\theta}{2}\right)\boldsymbol{n}$$
$$\boldsymbol{R}_s = \cos\left(\tfrac{\theta}{2}\right) + \sin\left(\tfrac{\theta}{2}\right)\boldsymbol{l} \text{ where } \boldsymbol{l} = \boldsymbol{n} + I\boldsymbol{m}$$
$$\boldsymbol{M} = \boldsymbol{T}_s\boldsymbol{R}_s = \left(1 + I\tfrac{\boldsymbol{t}_s}{2}\right)e^{\boldsymbol{l}\frac{\theta}{2}} = e^{\boldsymbol{l}\frac{\theta}{2} + I\frac{\boldsymbol{t}_s}{2}}$$
$$\boldsymbol{M} = \boldsymbol{T}_s\boldsymbol{R}_s = \left(1 + I\tfrac{\boldsymbol{t}_s}{2}\right)\mathbf{R}_s = \boldsymbol{R}_s + I\tfrac{\boldsymbol{t}_s}{2}\boldsymbol{R}_s = \boldsymbol{R}_s + I\boldsymbol{R}'_s$$
$$\boldsymbol{M} = \cos\left(\tfrac{\theta}{2} + I\tfrac{d}{2}\right) + \sin\left(\tfrac{\theta}{2} + I\tfrac{d}{2}\right)\boldsymbol{l}$$
$$|\boldsymbol{M}| = \boldsymbol{M}\widetilde{\boldsymbol{M}} = 1$$
$$\boldsymbol{R}_s\widetilde{\boldsymbol{R}}_s = 1,\ \ \widetilde{\boldsymbol{R}}_s\boldsymbol{R}'_s + \widetilde{\boldsymbol{R}}'_s\boldsymbol{R}_s = 0,\ \ \boldsymbol{t}_s = 2\boldsymbol{R}'_s\widetilde{\boldsymbol{R}}_s$$
$$\boldsymbol{M} = (a_0 + \boldsymbol{a}) + I(b_0 + \boldsymbol{b}) = \boldsymbol{T}_s\boldsymbol{R}_s$$
$$\widetilde{\boldsymbol{M}} = (a_0 - \boldsymbol{a}) + I(b_0 - \boldsymbol{b}) = \widetilde{\boldsymbol{R}}_s\widetilde{\boldsymbol{T}}_s$$
$$\bar{\boldsymbol{M}} = (a_0 + \boldsymbol{a}) - I(b_0 + \boldsymbol{b}) = \boldsymbol{R}_s\widetilde{\boldsymbol{T}}_s$$
$$\bar{\widetilde{\boldsymbol{M}}} = (a_0 - \boldsymbol{a}) - I(b_0 - \boldsymbol{b}) = \widetilde{\boldsymbol{R}}_s\boldsymbol{T}_s$$
$$a_0 = \tfrac{1}{4}(\boldsymbol{M} + \widetilde{\boldsymbol{M}} + \bar{\boldsymbol{M}} + \bar{\widetilde{\boldsymbol{M}}})$$
$$Ib_0 = \tfrac{1}{4}(\boldsymbol{M} + \widetilde{\boldsymbol{M}} - \bar{\boldsymbol{M}} - \bar{\widetilde{\boldsymbol{M}}})$$
$$a = \tfrac{1}{4}(\boldsymbol{M} - \widetilde{\boldsymbol{M}} + \bar{\boldsymbol{M}} - \bar{\widetilde{\boldsymbol{M}}})$$
$$I\boldsymbol{b} = \tfrac{1}{4}(\boldsymbol{M} - \widetilde{\boldsymbol{M}} - \bar{\boldsymbol{M}} + \bar{\widetilde{\boldsymbol{M}}})$$

Linear Algebra

$$\mathtt{f}(\boldsymbol{x}\wedge\boldsymbol{y}\wedge\cdots\wedge\boldsymbol{w}) = \mathtt{f}(\boldsymbol{x})\wedge\mathtt{f}(\boldsymbol{y})\wedge\cdots\wedge\mathtt{f}(\boldsymbol{w})$$
$$\mathtt{f}(I) = \mathtt{det(f)}I$$
$$\bar{\mathtt{f}}(\boldsymbol{y}) = e_k\mathtt{f}(e^k)\cdot\boldsymbol{y} = \partial_x\mathtt{f}(\boldsymbol{x})\cdot\boldsymbol{y}$$

$$\mathtt{f}(\boldsymbol{X}_r)\cdot\boldsymbol{Y}_s=\mathtt{f}[\boldsymbol{X}_r\cdot\bar{\mathtt{f}}(\boldsymbol{Y}_s)]\quad r\geq s$$
$$\boldsymbol{X}_r\cdot\bar{\mathtt{f}}(\boldsymbol{Y}_s)=\bar{\mathtt{f}}[\mathtt{f}(\boldsymbol{X}_r)\cdot\boldsymbol{Y}_s]\quad r\leq s$$
$$\mathtt{f}^{-1}(\boldsymbol{X})=\mathtt{det}(\mathtt{f})^{-1}\boldsymbol{I}\bar{\mathtt{f}}(\boldsymbol{I}^{-1}\boldsymbol{X})$$
$$\bar{\mathtt{f}}^{-1}(\boldsymbol{X})=\mathtt{det}(\mathtt{f})^{-1}\boldsymbol{I}\mathtt{f}(\boldsymbol{I}^{-1}\boldsymbol{X})$$

Frames and Contractions

$$e^i\cdot e_j=\delta^i_j$$
$$e^k=(-1)^{k+1}e_1\wedge\cdots\wedge\check{e}_k\wedge\cdots\wedge e_nE_n^{-1}$$
$$E_n=e_1\wedge e_2\wedge\cdots\wedge e_n$$
$$e_ke^k\cdot\boldsymbol{X}_r=\partial_x\boldsymbol{x}\cdot\boldsymbol{X}_r=r\boldsymbol{X}_r$$
$$e_ke^k\wedge\boldsymbol{X}_r=\partial_x\boldsymbol{x}\wedge\boldsymbol{X}_r=(n-r)\boldsymbol{X}_r$$
$$e_k\boldsymbol{X}_re^k=\dot{\partial}_x\boldsymbol{X}_r\dot{\boldsymbol{x}}=(-1)^r(n-2r)\boldsymbol{X}_r$$

Geometric Calculus

$$\nabla(\boldsymbol{XY})=\nabla\boldsymbol{XY}+\dot{\nabla}\boldsymbol{X}\dot{\boldsymbol{Y}}$$
$$\nabla\wedge\nabla=0$$
$$\nabla(\boldsymbol{x}\cdot\boldsymbol{y})=\boldsymbol{y}$$
$$\nabla\boldsymbol{x}^2=2\boldsymbol{x}$$

Incidence Algebra

$$PI^{-1}=\alpha II^{-1}=\alpha\equiv[P]$$
$$[\boldsymbol{x}_1\boldsymbol{x}_2\boldsymbol{x}_3\cdots\boldsymbol{x}_n]=[\boldsymbol{x}_1\wedge\boldsymbol{x}_2\wedge\boldsymbol{x}_3\wedge\cdots\wedge\boldsymbol{x}_n]=(\boldsymbol{x}_1\wedge\boldsymbol{x}_2\wedge\boldsymbol{x}_3\wedge\cdots\wedge\boldsymbol{x}_n)I^{-1}$$
$$P=\mathbf{X}_1\wedge\mathbf{X}_2\wedge\mathbf{X}_3\wedge\mathbf{X}_4=W_1W_2W_3W_4\langle(1+\boldsymbol{x}_1)(1-\boldsymbol{x}_2)(1+\boldsymbol{x}_3)(1-\boldsymbol{x}_4)\rangle_4$$
$$A^*=AI^{-1}$$
$$A\cdot B^*=(A\wedge B)^*=[A\wedge B]$$
$$J=A\cup B=A\wedge B$$
$$(A\cap B)^*=A^*\cup B^*\text{ for If }A=A'C\text{ and }B=B'C$$
$$A\cap B=(A^*\cdot B)$$

Direct Distance

$$\overline{e}\cdot A^h=\overline{e}\cdot(a_1^h\wedge a_2^h\wedge\cdots\wedge a_k^h)=(a_2-a_1)\wedge(a_3-a_2)\wedge\cdots\wedge(a_k-a_{k-1})$$
$$d[a_1^h\wedge\cdots a_k^h,b^h]\equiv[\{\overline{e}\cdot(a_1^h\wedge\cdots\wedge a_k^h)\}\quad(\overline{e}\cdot b^h)]^{-1}[\overline{e}\cdot(a_1^h\wedge\cdots\wedge a_k^h\wedge b^h)]=$$
$$[(a_2-a_1)\wedge\cdots\wedge(a_k-a_{k-1})]^{-1}[(a_2-a_1)\wedge\cdots\wedge(a_k-a_{k-1})\wedge(b-a_k)]$$
$$d[a_1^h\wedge\cdots\wedge a_r^h,b_1^h\wedge\cdots\wedge b_s^h]\equiv\{\overline{e}\cdot(a_1^h\wedge\cdots\wedge a_r^h)\}\wedge\{\overline{e}\cdot(b_1^h\wedge b_2^h\wedge\cdots$$
$$\cdots\wedge b_s^h)\}]^{-1}\qquad[\overline{e}\cdot(a_1^h\wedge\cdots\wedge a_r^h\wedge b_1^h\wedge\cdots\wedge b_s^h)]=[(a_2-a_1)\wedge\cdots\wedge(a_r-$$
$$a_{r-1})\wedge(b_2-b_1)\wedge\cdots\wedge\qquad(b_s-b_{s-1})]^{-1}[(a_2-a_1)\wedge\cdots\wedge(a_r-a_{r-1})\wedge(b_1-$$
$$a_r)\wedge(b_2-b_1)\wedge\cdots\wedge(b_s-b_{s-1})]$$

Projective Geometry

$$\mathbf{X}\gamma_{n+1} = \mathbf{X}\cdot\gamma_{n+1} + \mathbf{X}\wedge\gamma_{n+1} = \mathbf{X}\cdot\gamma_{n+1}\left(1+\frac{\mathbf{X}\wedge\gamma_{n+1}}{\mathbf{X}\gamma_{n+1}}\right) \qquad \text{projective split}$$

$$X\gamma_4 = \mathbf{X}\cdot\gamma_4 + \mathbf{X}\wedge\gamma_4 = X_4\left(1+\frac{\mathbf{X}\wedge\gamma_4}{X_4}\right) \equiv X_4(1+\boldsymbol{x})$$

$$L\cap\Phi = (\mathbf{X}_1\wedge\mathbf{X}_2)\cap(\mathbf{Y}_1\wedge\mathbf{Y}_2\wedge\mathbf{Y}_3) = L^*\cdot\Phi$$

$$L\cap\Phi = [\mathbf{X}_1\mathbf{X}_2\mathbf{Y}_2\mathbf{Y}_3]\mathbf{Y}_1 + [\mathbf{X}_1\mathbf{X}_2\mathbf{Y}_3\mathbf{Y}_1]\mathbf{Y}_2 + [\mathbf{X}_1\mathbf{X}_2\mathbf{Y}_1\mathbf{Y}_2]\mathbf{Y}_3$$

$L_1\wedge L_2 = 0$ coplanar lines

$L_1\cap L_2 = [\mathbf{X}_1\mathbf{X}_2\mathbf{Y}_1]\mathbf{Y}_2 - [\mathbf{X}_1\mathbf{X}_2\mathbf{Y}_2]\mathbf{Y}_1$ intersecting lines

$$L = \Phi_1\cap\Phi_2 = (\mathbf{X}_1\wedge\mathbf{X}_2\wedge\mathbf{X}_3)\cap(\mathbf{Y}_1\wedge\mathbf{Y}_2\wedge\mathbf{Y}_3) = [\mathbf{X}_1\mathbf{X}_2\mathbf{X}_3\mathbf{Y}_1](\mathbf{Y}_2\wedge\mathbf{Y}_3) +$$
$$+ [\mathbf{X}_1\mathbf{X}_2\mathbf{X}_3\mathbf{Y}_2](\mathbf{Y}_3\wedge\mathbf{Y}_1) + [\mathbf{X}_1\mathbf{X}_2\mathbf{X}_3\mathbf{Y}_3](\mathbf{Y}_1\wedge\mathbf{Y}_2)$$

Projective Invariants

$$Inv_1 = \frac{(\mathbf{X}_3\wedge\mathbf{X}_1)I_2^{-1}(\mathbf{X}_4\wedge\mathbf{X}_2)I_2^{-1}}{(\mathbf{X}_4\wedge\mathbf{X}_1)I_2^{-1}(\mathbf{X}_3\wedge\mathbf{X}_2)I_2^{-1}} = \frac{(t_3-t_1)(t_4-t_2)}{(t_4-t_1)(t_3-t_2)}$$

$$Inv_2 = \frac{(\mathbf{X}_5\wedge\mathbf{X}_4\wedge\mathbf{X}_3)I_3^{-1}(\mathbf{X}_5\wedge\mathbf{X}_2\wedge\mathbf{X}_1)I_3^{-1}}{(\mathbf{X}_5\wedge\mathbf{X}_1\wedge\mathbf{X}_3)I_3^{-1}(\mathbf{X}_5\wedge\mathbf{X}_2\wedge\mathbf{X}_4)I_3^{-1}} = \frac{A_{543}A_{521}}{A_{513}A_{524}}$$

$$Inv_3 = \frac{(\mathbf{X}_1\wedge\mathbf{X}_2\wedge\mathbf{X}_3\wedge\mathbf{X}_4)I_4^{-1}(\mathbf{X}_4\wedge\mathbf{X}_5\wedge\mathbf{X}_2\wedge\mathbf{X}_6)I_4^{-1}}{(\mathbf{X}_1\wedge\mathbf{X}_2\wedge\mathbf{X}_4\wedge\mathbf{X}_5)I_4^{-1}(\mathbf{X}_3\wedge\mathbf{X}_4\wedge\mathbf{X}_2\wedge\mathbf{X}_6)I_4^{-1}} = \frac{V_{1234}V_{4526}}{V_{1245}V_{3426}}$$

Conformal Geometric Algebra

$e_i^2 = 1,\ i = 1,\ldots,n;\ \ e_0 = \frac{(e_- - e_+)}{2},\ \ e_\infty = e_- + e_+$

$E = e_\infty\wedge e_0 = e_+\wedge e_- = e_+e_-$ Minkowski plane

$\boldsymbol{x}_c = \mathbf{x}_e + \alpha e_0 + \beta e_\infty$ conformal split

$P_E(\boldsymbol{x}_c) = (\boldsymbol{x}_c\cdot\boldsymbol{E})\boldsymbol{E} = \alpha e_0 + \beta e_\infty \in \mathbb{R}^{1,1}$

$P_E^\perp(\boldsymbol{x}_c) = (\boldsymbol{x}_c\cdot\boldsymbol{E}^*)\widetilde{\boldsymbol{E}^*} = (\boldsymbol{x}_c\wedge\boldsymbol{E})\boldsymbol{E} = \mathbf{x}_e \in \mathbb{R}^n$

$\boldsymbol{x}_c = P_E(\boldsymbol{x}_c) + P_E^\perp(\boldsymbol{x}_c).$

$\boldsymbol{x}_c = \boldsymbol{x}_c\boldsymbol{E}^2 = (\boldsymbol{x}_c\wedge\boldsymbol{E} + \boldsymbol{x}_c\cdot\boldsymbol{E})\boldsymbol{E} = (\boldsymbol{x}_c\wedge\boldsymbol{E})\boldsymbol{E} + (\boldsymbol{x}_c\cdot\boldsymbol{E})\boldsymbol{E}$

$\boldsymbol{x}_c = (\boldsymbol{x}_c\wedge\boldsymbol{E})\boldsymbol{E} + (\boldsymbol{x}_c\cdot\boldsymbol{E})\boldsymbol{E} = \mathbf{x}_e + e_0 + \frac{1}{2}(k_1+k_2)e_\infty = \mathbf{x}_e + \frac{1}{2}\mathbf{x}_e^2 e_\infty + e_0$

$\boldsymbol{x}_c = \mathbf{x} + \frac{1}{2}\mathbf{x}^2 e_\infty + e_0$ point

$\boldsymbol{x}_c^* = \boldsymbol{s}_1\wedge\boldsymbol{s}_2\wedge\boldsymbol{s}_3\wedge\boldsymbol{s}_4$ dual point

$\boldsymbol{s} = \mathbf{p} + \frac{1}{2}(\mathbf{p}^2-\rho^2)e_\infty + e_0$ sphere

$\boldsymbol{s}^* = \boldsymbol{a}\wedge\boldsymbol{b}\wedge\boldsymbol{c}\wedge\boldsymbol{d}$ dual sphere

$\boldsymbol{x}_c\wedge\boldsymbol{s}^* = 0$

$\pi = \boldsymbol{n}I_E - de_\infty\boldsymbol{n} = (\boldsymbol{a}-\boldsymbol{b})\wedge(\boldsymbol{a}-\boldsymbol{c})\ \ d = (\boldsymbol{a}\wedge\boldsymbol{b}\wedge\boldsymbol{c})I_E$ plane

$\pi^* = e_\infty\wedge\boldsymbol{a}\wedge\boldsymbol{b}\wedge\boldsymbol{c}$ dual plane

$\boldsymbol{L} = \pi_1\wedge\pi_2,\ \boldsymbol{L} = \boldsymbol{n}I_E - e_\infty\boldsymbol{m}I_E\ \ \boldsymbol{n} = (\boldsymbol{a}-\boldsymbol{b}),\ \ \boldsymbol{m} = (\boldsymbol{a}\wedge\boldsymbol{b})$ line

$\boldsymbol{L}^* = e_\infty\wedge\boldsymbol{a}\wedge\boldsymbol{b}$ dual line

$\boldsymbol{z} = \boldsymbol{s}_1\wedge\boldsymbol{s}_2 = \boldsymbol{s}_1\wedge\pi_2$, circle

$\boldsymbol{z}^* = \boldsymbol{a}\wedge\boldsymbol{b}\wedge\boldsymbol{c}$, dual circle

$\boldsymbol{P}\boldsymbol{P} = \boldsymbol{s}_1\wedge\boldsymbol{s}_2\wedge\boldsymbol{s}_3,\ \ \boldsymbol{P}\boldsymbol{P} = \boldsymbol{s}\wedge\boldsymbol{L}$ point pair

$\boldsymbol{P}\boldsymbol{P}^* = \boldsymbol{a}\wedge\boldsymbol{b}$ dual point pair

$\boldsymbol{s} = \boldsymbol{A}_r\boldsymbol{A}_{r+1}^{-1}$

$\boldsymbol{s} = \boldsymbol{z}\pi^{-1},\ \ \boldsymbol{s} = \boldsymbol{P}\boldsymbol{P}\boldsymbol{L}^{-1},\ \ \boldsymbol{P}\boldsymbol{P} = \boldsymbol{s}\boldsymbol{L} = \boldsymbol{s}\wedge\boldsymbol{L}$

Lie Algebra of the Conformal Group

$$[\boldsymbol{x} \wedge \boldsymbol{y} - (\boldsymbol{x} \cdot \boldsymbol{E})(\boldsymbol{y} \cdot \boldsymbol{E})] \times \boldsymbol{E} = 0$$
$$\boldsymbol{E}_i = e_+ e_-,$$
$$\boldsymbol{B}_{ij} = e_i e_j \qquad (i < j = 1, \ldots, n),$$
$$\bar{\boldsymbol{E}}_{ij} = e_i e_\pm$$

Conformal Transformations

$s(\boldsymbol{x}_c) = -\boldsymbol{s}\boldsymbol{x}_c\boldsymbol{s}^{-1}$ inversion
$\boldsymbol{x}' = -\pi\boldsymbol{x}\pi^{-1}$ reflection
$\boldsymbol{K}_{\mathbf{b}} = e_+\boldsymbol{T}_{\mathbf{b}}e_+ = (e_\infty - e_0)(1 + \mathbf{b}e_\infty)(e_\infty - e_0) = 1 + \mathbf{b}e_0$ transversions
$\boldsymbol{T}_a = 1 + \frac{1}{2}ae_\infty = e^{-\frac{a}{2}e_\infty}$ translator
$\boldsymbol{R}_\theta = n_2 n_1 = \cos\left(\frac{\theta}{2}\right) - \sin\left(\frac{\theta}{2}\right) l = e^{-\frac{\theta}{2}l}$ rotor
$\boldsymbol{M}_\theta = \boldsymbol{T}\boldsymbol{R}\widetilde{\boldsymbol{T}} = \cos\left(\frac{\theta}{2}\right) - \sin\left(\frac{\theta}{2}\right) L = e^{-\frac{\theta}{2}L}$ motor
$Q' = \prod_{i=1}^{n} M_i \, Q \prod_{i=1}^{n} \widetilde{M}_{n-i+1}$
$\boldsymbol{D}_\rho = (1 + \boldsymbol{E})\rho + (1 - \boldsymbol{E})\rho^{-1} = e^{\boldsymbol{E}\phi}$, dilator
$\boldsymbol{E}(\mathbf{x}_e + \mathbf{x}_e^2 e_\infty + e_0)\boldsymbol{E} = -(-\mathbf{x}_e + \frac{1}{2}\mathbf{x}_e^2 e_\infty + e_0)$, involution
$\boldsymbol{G} = \boldsymbol{K}_{\mathbf{b}}\boldsymbol{T}_{\mathbf{a}}\boldsymbol{R}_\alpha$ conformal transformation
$g(\boldsymbol{x}_c) = \boldsymbol{G}\boldsymbol{x}_c(\boldsymbol{G}^*)^{-1} = \sigma\boldsymbol{x}_c'$

Differential Kinematics

$$x_p' = \prod_{i=1}^{n} M_i x_p \prod_{i=1}^{n} \widetilde{M}_{n-i+1}$$
$$dx_p' = \sum_{j=1}^{n} \left[x_p' \cdot L_j'\right] dq_j$$

Dynamics

$$\mathcal{M}\ddot{q} + C\dot{q} + G = \tau$$
$$\mathcal{M} = V^T m V + \delta I$$
$$C = V^T m \dot{V}$$
$$G = V^T m \dot{V} = V^T m a$$
$$V = \begin{pmatrix} x_1' & 0 & \cdots & 0 \\ 0 & x_2' & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & x_n' \end{pmatrix} \begin{pmatrix} L_1' & 0 & \cdots & 0 \\ L_1' & L_2' & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ L_1' & L_2' & \cdots & L_n' \end{pmatrix} = XL$$
$$(V^T m V + \delta I)\ddot{q} + V^T m \dot{V}\dot{q} + V^T F = \tau$$
$$\delta I\ddot{q} + V^T(mV\ddot{q} + m\dot{V}\dot{q} + F) = \tau$$
$$\delta I\ddot{q} + V^T m(V\ddot{q} + \dot{V}\dot{q} + a) = \tau$$

Index

A
Abelian subgroup, 75
Additive split, 182, 229, 230
Affine
– , n-dimensional plane, 94
– n-plane, 102
– plane, 104, 197, 198, 287
– transformation, 84
Algebra
– , bivector, 75
– of Clifford, 23
– of Grassmann, 2, 23
Algebraic generators, 83
Algorithm
– RANSAC, 511
– , TPS-RPM, 558
Alignment condition, 276
Angle dihedral, 149
Angular momentum, 166, 299
Anticommutator product, 30
Antisymmetric part, 29
Attitude condition, 276
Automorphism, 144

B
Backward transformation, 274
Basis, Witt, 64
Bi-quaternion, 126, 138
Bivector, 23, 30, 126, 129, 131
– , doubling, 60, 199
– , involutory, 79
Bivector algebra, 56
Blade, 31
– , magnitude or modulus of a, 42
– of grade r, 24
Body–eye
– calibration problem, 470
Body–eye calibration algorithm, 507
Bracket, 119

C
Canonical homomorphism, 81
Cayley
—Menger determinant, 197
Clifford
– algebra, 23
– , convolution, 46
– , correlation, 46
Clifford algebra, 2
Closed-loop system, 365
Co-adjoint, 227
Cognitive systems, 22
Collision avoidance, 415
Common dividend of lowest grade, 43
Commutator product, 30
– of bivectors, 68
Complex
– number, 59, 585
– structure, 63, 73
Compliance, 305
Composed number, 126
Conformal
– , direct kinematics, 206
– geometric algebra, 181, 219
– , geometry, 187
– , main involution, 208
– , mapping, 197
– , spaces, 185
– , split, 182, 186
– , transformation, 202, 209
Contraction, 37

© Springer Nature Switzerland AG 2020

E. Bayro-Corrochano, *Geometric Algebra Applications Vol. II*,
https://doi.org/10.1007/978-3-030-34978-3

Control
– Bang–Bang, 369
– , sliding mode, 363
Controller
– , Bang–Bang, 367
– , computed-torque proportional-derivative, 371
– , integral sliding modes, 423
– , PD, 365
– PD-CTC, 371
– , sliding mode, 366
– super twisting, 421
Convolution, 45
Coordinates
– of *Plücker*, 82
Coriolis
– tensors, 356
– theorem, 312
Co-screw, 221, 225, 237, 361
– momenta, 227
Cross product, 40
CT image, 555
CUDA, 261
Cylinder, 232

D
Damping velocity, 415
De Casteljou algorithm, 258, 259
Degenerated motor, 138
Delaunay tetrahedralization, 547
Denavit–Hartenberg
– parameters, 271, 279
– procedure, 271
Deterministic annealing, 559
Diagonal inertia, 226
Differential kinematics, 171
Dilator, 93, 208
Directance, 184
Directed distance, 111, 199
– of lines, 111
– of planes, 112
Direction
– moment, 199
Direct kinematics, 272, 277
Direct product, 585
Directrix, 209
Double-cover representation, 74
Double motor, 236
Double motor algebra, 220, 237
Doubling
– bivector, 60, 115
– bivector $\boldsymbol{J}$, 73
Dual
– angles, 273
– , equation of the point, 156
– , function of a variable, 125
– null cone, 237
– number, 126, 127
– spinor representation, 141
– terms of direction and moment, 140
– , unit shear number, 127
Duality, 40
Dual Quaternion Iterative Blending, 256
Dynamic equation, 167
Dynamic motion model, 449
Dynamics
– of a robot arm, 334

E
Eigenvalue, 77
Eigenvector, 77
Ellipsoid, 231
End-effector, 272
End-effector frame, 275
Endomorphism, 96
Endoscopes in surgeries, 473
Endoscopic camera, 469
Energy
– , kinetic, 321
Enforcing geometric constraint, 452
Estimation
– , batch using SVD techniques, 432
– of 3D Euclidean transformation, 431
– of 3D rigid motion using 3D line observations, 441
– of the hand–eye motor using SVD, 436
– rotation, 447
Euclidean
– , covariant geometry, 183
– plane, 126
Euler–Lagrange equations, 356
Euler representation, 139
Euler's equation, 312
Even subalgebra, 138
Exponential mapping, 97
Extended Kalman filter, 443
– , motor, 448
– rotor, 445

F
Flag, 114, 178, 585
– , involving points, lines and planes, 174
– line–plane, 174
– point-line, 174

– point–line–plane, 175
– soma, 206
Forward kinematics, 393
Forward transformation, 274
Frame
– end-effector, 275
– , nonorthonormal, 26
– , reciprocal with curvilinear coordinates, 27
Frobenius theorem, 135

G
Generalized coordinates, 368
Generalized dot product, 100
Generalized homogeneous coordinates, 186
Generalized inner product, 31, 34, 38
General linear Lie algebra, 99
Generation of trajectories, 413
Geometric
– indicator, 174
– product, 24, 36, 51
Geometric algebra, 2, 23
– , degenerated, 270
– , Euclidean, 129
Geometric constraint, 452
Geometric product, 131
Globally asymptotically stable, 363
Grade involution, 42
Grasping operation, 275
Grassmann
– algebra, 65, 74, 81
– , reciprocal subalgebras, 63
Grid reference frame, 468
Group
– , Abelian, 75
– , complex orthogonal $SO(n, \mathbb{C})$, 79
– , double covering, 221
– , general linear, GL(n, $\mathbb{R}$), 78
– , orthogonal SO(2), 100
– , Pin^{sp}(3,3), Pin(3,3), 81
– , Pin(n), 80
– , rotor, 56
– , $SE(3)$, 219
– , $SL(4)$, 220
– , $SL(4)$, $Spin$(3,3), 81
– , SO(n), 80
– , special linear, SL(n, $\mathbb{R}$), 78
– , special unitary $SU(n)$, 63
– , Spin(n), 80
– , Spin(n, n), 81
– , spinor Spin(2), 101
– , sub-one-parameter, 104
– , unitary, 61
– , unitary $U(n)$, $U(p, q)$, 62

H
Hamiltonian, 372
Hamilton, W.R., 135
Hand–eye
– , using CGA, 520
Hand–eye calibration
– in CGA, 465
Hand–eye problem, 432
– Euclidean transformation, 432
Hermitian
– , inner product, 61
– , quaternionic function, 137
Hesse distance, 154, 156, 199, 286
Hodge
– , dual, 39
– star operator, 39
Hodgkin–Huxley, 380
Homogeneous
– coordinates, 108, 137, 147
– representant, 110
– subspaces, 24
– vector, 24
Homogeneous coordinates, 270
Homography, 84
Horosphere, 94, 103, 186
– , homogeneous model, 186
Hough transform
– for line detection, 526
Hyperplane, 185
Hypersphere, 187
Hypervolume, 126

I
Idempotent, 71
Incidence operator, 40
Incidence relation
– , point, line, and a plane, 174
Inertia
– , diagonal, 226
– matrix, 222, 225, 238
– , time derivative of, 310
Inertia matrix, 301, 303, 306
Inertia of the link, 323
Inertor, 239
Inner product, 29
– , generalized, 31
– , Hermitian, 61
– or contraction, 36

Inner Product Null Space, 189
Intersection, 109
– of surfaces, 234
– of two ellipsoids, 233
Invariance, 224
Invariant, 225
– $\boldsymbol{J}$, 62
Invariant, quadratic form, 145
Inverse dynamics, 418
Inverse kinematic, 272, 279, 287
– using conformal geometric algebra, 290
– using motor algebra, 279
– using the 3D affine plane, 287
Inversion, 202
Involution, 208
IPNS Inner Product Null Space, 190
IPNS representation
– , of the line, 192
– , of the point, line, plane, circle, sphere, 193
– , of the sphere, 190
Iterated Closest Point (ICP), 558

J
Jacobian, 306, 414
Jacobi identity, 62
Join, 43, 109
Joint, 271
Joint transition, 271

K
Kalman filter, 441
Killing form, 58
Kinematics
– , differential, 171
– direct, 277
– inverse, 279
Kinetic energy, 222, 226, 321
– of a robot arm, 323
k-vector, 24

L
Lagrange
– interpolating polynomials, 256
Laser scanner, 507
LasSalle's theorem, 363
Lie
– , algebra generators, 56, 206
– algebras, 97
– algebra $u(p, q)$, 63
– , bivector algebra, 58
– bracket, 99, 221, 304
– , co-bracket, 305, 311
– generators, 86
– , group manifold, 56
– , group of rotors, 56, 58
– groups, 97
– group $SL(4, \mathbb{R})$, 86
– , group theory, 55
– operators, 85
Lie algebra, 55, 56, 199, 220
– generators, 75, 77
– , of the conformal group, 201
– $SL(4)$, 81
– $sl(4)$, 220
– $sl(n)$, 75
– $SO(n, n)$, 76
Lie group, 58, 70, 77, 220, 301
– $SL(4)$, 81
– $Spin(n, n)$, 76
– versors, 86
Line, 234
– direction, 155
– moment, 155
Linear algebra, 47
– , derivations, 48
Linear motion equation, 451
Line motion model, 449
Lobachevsky Nikolai, 185
Local Hamiltonians, 368
Localized control, 361
Logical cube, 548
Lorentor, 90, 235
Lorentz transformation, 89, 127, 128
Lyapunov function, 362

M
Magnitude, 42
– of a rotor, 130
Manifold, 55, 97
Manipulator
– SCARA, 271, 279
– Stanford, 271, 272, 280
Map 2D, 510
Map building, 510
– using laser and stereo vision, 519
Map building 3D, 512
Maple, 278, 282, 286, 321
Marching
– cubes, 547
– spheres, 547
Marching cubes algorithm, 548

Marching spheres
– , object registration, 555
Matching laser reading, 507
Matrix
– , Coriolis, 331
Measurement model, 444
Medical image analysis, 547
Meet, 43, 109, 289
– product, 223
MEKF filter, 454
Metric
– , Euclidean, 34
– geometry, 33
– , Minkowski, 34
Minkowski plane, 35, 182, 199
Model
– LIF, 381
– PIF, 381
Model of motion of lines in 4D space, 457
Modulus, 42
Momenta, 221, 225, 227, 237, 303, 361
– , time derivative of, 303
Motion
– , 3D model of the line, 157
– , 3D model of the plane, 157
– , 3D model of the point, 157
– , 4D model of the line, 160
– , 4D model of the plane, 161
– , 4D model of the point, 160
– equation, of a line, 170
– equation, of a plane, 170
– equation, of a point, 170
– , motor model of the line, 158
– , motor model of the plane, 159
– , motor model of the point, 158
– , screw of the line, 158
Motion planing, 413
Motor, 126, 138
– , degenerated, 142
– , dual part of the, 142
– interpolation, 255
– Iterative Blending, 256
– reflections, 144
– , transformation, 273
Motor algebra, 219
– , double, 220
– , spatial velocity, 168, 532
Motor Extended Kalman Filter, 448
Motor Spherical Linear Interpolation, 255
MSLERP, 255
Multi-streaming, 263
Multivalued function field, 45
Multivector, 24
– , dual of a, 40
– inhomogeneous, 36
– , magnitude of a, 42
– product, 29
– , valued function, 44

N
Neural network
– , quaternion spiking, 379
Neuroplasticity, 379
Neutral pseudo-Euclidean space, 64
Newton–Euler
– equation, 314
– recursive algorithm, 312, 319, 369, 418
Newton-Euler recursive algorithm, 369
Nonorthonormal frames, 26
Null
– basis, 182, 228, 237
– cone, 185, 186
– structure, 73
– vector, 182, 183, 187, 220, 228, 237
Null cones
– , reciprocal, 65
Null Space
– , Inner Product, 190
– , Outer Product, 190
Number
– , complex, 125, 127
– , composed, 125
– , dual, 125

O
Object registration, 555
Observable system, 34
Odometry, 509
Omnidirectional camera, 512
One-parameter group, 97
Operator
– , Hodge star, 39
OPNS
– , Outer Product Null Space, 190
OPNS representation
– , of the circle, 192
– , of the plane, 191
– , of the point, 193
– , of the point, line, plane, circle, sphere, 193
– , of the sphere, 191
Optimal Kalman gain matrix, 452
Outermorphism, 47
Outer Product Null Space, 189

P
Pair of points, 194, 234
Pan–Tilt unit, 525
Path following, 514
Perspector, 88, 235
Phase space, 370
Pitch, 138
Plane at infinity, 110, 233
Plücker coordinates, 155
Point at infinity, 220
Postsynaptic potential, 383
Potential energy, 328
Principle
– , D'Alembert's, 297
– , Hamilton, 298
– , Lagrange–d'Alembert, 297
– Pontryagin, 372
Product
– , anticommutator, 30
– , commutator, 30
– , cross, 40
– , generalized inner, 34
– , inner, 40
– , outer, 40
Projection, 32, 182
Projective
– geometry, 33
– rays, 108
– split, 33, 73, 147
– transformation, 235
Pseudoscalar, 40

Q
Quadratic programming, 413, 417
Quaternion, 131
– algebra, 129
– spike neural network, 391
– , unit, 129
Quaternion algebra, 585
Quaternion Spiking neural network, 379

R
Reciprocal null cones, 63, 65, 228
Reflection, 129, 130, 148
Registration methods, 558
Rejection, 33, 182
Rendezvous method, 280
Representation
– , 3D of the line, 153
– , 3D of the plane, 154
– , 3D of the point, 153
– , 4D of the line, 156
– , 4D of the plane, 156
– , 4D of the point, 156
– , asymmetric, 155
– , motor of the line, 155
– , motor of the plane, 155
– , motor of the point, 155
Reversion, 42, 130
Rigid body motion in CGA, 463
Rigid motion, 220, 270
Robot arm, 372
Robot navigation, 513
Robust point matching (TPS-RPM), 558
Rodriguez formula, 148, 164, 177
Rotation, orthogonal, 133
Rotator, 89
Rotor, 129–131, 133, 138, 139
– algebra, 129
– , algebra of, 131
– angular velocity, 166
– group, 56
– , orthogonal, 134
– , two conformal reflections, 205
– , unit, 131
Ruled surface, 209
– , cone and a sphere, 212
– , cycloidal curves, 210
– , helicoid, 211
– , Plücker conoid, 212

S
Screw, 138, 225, 271, 301
– axis line, 138
– , co-, 225
– line, 302
– motion, 302
– transformation, 273
Screw axis, 448
Screw motion, 255
Segmentation, 550
– , boundary and region based, 550
– , brain tumor, 552
– , texture, 550
Self-adjoint, 47
Sensor calibration, 507
Shearotor, 91
Sheartor, 235
Shear transformation, 235
Shuffle product, 223
Signature, 126, 147
– , Euclidean, 147
Simplex, 48, 111, 216

– and sphere, 196
– , operators, 50
Simpson rule, 214
SLERP, 255
Sliding mode control, 363
Sliding surface, 371
Space
– , anti-Euclidean, 68
– , neutral pseudo-Euclidean, 64
Space–time
– split, 34
– system, 34
Spatial velocity, 165
– , rigid body, 161
Special unitary group $SU(n)$, 63
Spectral form, 97
Sphere, 232
– , dual, 190
Spherical Linear Interpolation, 255
Spiking neural network, 379
Spin group, 127
Spinor, 138, 140
– , dual-number representation of the, 141
Split
– , additive, 183
– , conformal for points and simplexes, 183
– , multiplicative, 183
Stability
– condition, 76
– group, 76, 78
Stanford manipulator, 271, 280
Stereographic projection, 187, 229
Structure
– , complex, 73
– constants, 58
– , null, 73
Study
– kinematics mapping, 257
– manifold, 258
Study quadric, 257
– interpolation algorithm, 257
Symmetric part, 29
Symmetry, 326

T
Tangent operators, 99
Taylor
– expansion, 58, 60, 61
– series, 125, 141
Theorem
– , Apollonius, 214
– , Desargues, 120, 215
– , Pascal, 215
Touching condition, 276
Tracker
– endoscope, 463
– endoscope calibration, 467, 567
– Polaris, 468
Tracking, 366
Transformation, 126, 128, 132
– , backward, 274, 284
– , Euclidean, 137, 138
– , forward, 274, 284
– , group on the plane, 126
– , Lorentz, 89
– of points, lines, planes, 273
– , prismatic, 273, 274
– , revolute, 273, 274
– , shear, 127
Translation, 202
Translator, 87, 139, 140, 274
Transversor, 205
Trivector, 30, 129
Twist, 138, 303
– , coupled, 209
– , exponential, 164

U
Union, 109
Union of Spheres, 548
Unitary group, 61
Unitary groups $U(n)$, $U(p, q)$, 62
Unit sphere, 585
Universal geometric algebra, 65

V
Vector
– , bases reciprocal, 64
– field, 107
– , orthogonal null, bases, 68
– , orthonormal, bases, 68
Velocity
– , linear and angular, 165
Versors, 102, 202, 228, 235
– , for dilation, 208
– , for inversion, 203
– , for transversion, 205
– , representation, 202
Vertex, 209
Virtual work, 298

Visual invariants, 103

W
Wedge product, 29
Widrow–Hoff rule, 386
Witt basis, 64, 65, 94
Wrench, 303, 306